Science of Computer Programming 74 (2009) 498–533

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

The logic of message-passing☆

J.R.B. Cockett [a,*], Craig Pastro [b]

[a] Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, Alberta T2N 1N4, Canada
[b] Department of Mathematics, Macquarie University, NSW 2109, Australia

ARTICLE INFO

Article history:
Received 23 March 2007
Received in revised form 20 September 2007
Accepted 23 November 2007
Available online 11 February 2009

Keywords:
Message passing
Concurrency
Process semantics
Linear logic
Term logic
Multicategory
Polycategory
Linear distributive category
Poly-actegory
Linear actegory

ABSTRACT

Message-passing is a key ingredient of concurrent programming. The purpose of this paper is to describe the equivalence between the proof theory, the categorical semantics, and term calculus of message-passing. In order to achieve this we introduce the categorical notion of a linear actegory and the related polycategorical notion of a poly-actegory. Not surprisingly the notation used for the term calculus borrows heavily from the (synchronous) π-calculus. The cut-elimination procedure for the system provides an operational semantics.

1. Introduction

If programs should be viewed as proofs and types as propositions then for what proof system are *concurrent* programs the proofs? Sequential programs are connected through the λ-calculus by the Curry-Howard-Lambek isomorphism to intuitionistic proofs and cartesian closed categories. Considering the impact that connection has had on our understanding of sequential programs, it is reasonable to suppose that the answer to the question for concurrent programs might have similar and far-reaching consequences.

On the face of it the question seems beguilingly easy. Indeed, one might reasonably be tempted to backward-engineer the π-calculus [22,23] – which after all holds in the concurrent world an analogous position to the λ-calculus [7] – to arrive at an answer. However, a moment's thought about the passage between the λ-calculus and proofs makes one realise that the significant missing component, namely the type system, has a huge effect. Adding types to the λ-calculus introduces a program discipline which, for example, is sufficiently restrictive to provide a guarantee of termination. This carries the λ-calculus far from its freewheeling role as a description of computability.

☆ The first author gratefully acknowledges the support of NSERC, Canada while the second gratefully acknowledges the support of an international Macquarie University Research Scholarship. Parts of this work were completed while the second author was visiting the University of Calgary during which he was also supported by the Department of Computer Science at the University of Calgary, Macquarie International, and the ICS Postgraduate Research Fund.

* Corresponding author. Tel.: +1 403 220 5106; fax: +1 403 284 4707.
E-mail addresses: robin@cpsc.ucalgary.ca (J.R.B. Cockett), craig@maths.mq.edu.au (C. Pastro).

0167-6423/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2007.11.005

Concurrent programs are even more freewheeling than sequential programs and, therefore, it is inevitable that collecting them into a proof system will introduce a programming discipline which has a similar effect of guaranteeing some strong formal properties (for example, being deadlock and livelock free). Unlike the λ-calculus, however, whose development was concomitant with the philosophical underpinnings of type theory, the π-calculus was developed in a brave new world of computing in which operational behaviours sufficed. Thus, the connection to a proof system did not seem inevitable or even necessary.

Despite this it would be a strange world indeed if there was no type-theoretic underpinning to concurrent programming. There is now a considerable body of literature connecting concurrent semantics with linear logic – or at least the multiplicative fragment of it embodied by the logic of the polycategorical cut. While this perhaps should not be viewed as providing conclusive evidence that linear logic is the correct basis in proof theory for concurrent semantics, the argument at this stage is quite compelling. See for example the work of Abramsky on interaction categories [2,3], of Abramsky and Melliès [5], of Barber, Gardner, Hasegawa, and Plotkin [6], and our own work [13,24]. In particular, the authors were influenced by the much earlier paper of Bellin and Scott [8] in which this connection was pursued and which contains further historical commentary.

So how does one model message-passing in this formal setting? Well, of course there is a technical answer to that question which covers the pages which follow. However, we should draw the reader's attention to one particular aspect of some consequence. We model message-passing using a two-tier logic. There is a logic for the messages whose proofs should be thought of as ordinary sequential programs. Then there is a logic of message-passing which is built on top of the logic of messages. The two logics are really quite distinct: the message logic is concerned with what we classically view as computation, while the second logic is concerned with manipulating the channels of communication.

We believe there is a – perhaps somewhat uncomfortable – message in this. Both functional and imperative programming language designers have introduced concurrent features, essentially, by either adding operating system primitives to their sequential core or by overloading basically sequential constructs (consider the use of monads to obtain IO in Haskell). However, even the briefest perusal of the rules, indicates that the logic concerned with managing channels is at least as complicated as the sequential programming logic. Furthermore, there are quite significant interactions between the two levels. This suggests that trying to place a boundary to programming language design at this point is altogether artificial. Thus, we believe programming language designers *should* be thinking in terms of developing integrated two tier languages in order to give high-level support for concurrent programming — operating system primitives manifestly fail in this regard.

In this paper we do not pretend that a logic whose proofs are concurrent programs is going to be a particularly simple thing. There are many rules involved and consequently many equivalences between proofs. However, there is nothing dramatically original about the proof theory we present either. It is basically the proof theory of (polycategorical) cut (see [10]) with messages. The aim of this paper is to lay out in some detail how the expression of message-passing is added to the logic. The resulting system is necessarily somewhat more complex as it has to include the logic of the messages themselves. The logic for message-passing is then built on top of the logic for messages and embodies the interactions which are necessary between the two levels.

In order to provide a basis which would cover a broad range of semantics we decided to use a logic of messages whose categorical semantics is a distributive monoidal category. That is a monoidal category with coproducts over which the tensor distributes. Explicitly this means that the natural map

$$A \otimes X + A \otimes Y \xrightarrow{\langle 1_A \otimes \mathrm{inj}_l \,,\, 1_A \otimes \mathrm{inj}_r \rangle} A \otimes (X + Y)$$

is an isomorphism. The presence of coproducts in the messages allows us to indicate how this structure must interact with the message-passing level. This is a rather crucial aspect of the system we present which allows the contents of messages to determine the interaction which actually unfolds.

In order to illustrate this point consider the following simple program which runs a bank machine. A user will insert their card into the bank machine and provide their personal identification number, *pin*, and a request for money, *x*. The bank machine will send this information to the bank which will respond with a transaction identification number, *tid*, and an amount, *y*, which the bank has permitted the bank machine to deliver to the user. This amount may be either the amount the user has requested or, in the case the user had made a request which cannot be satisfied (e.g., the request will put them over their daily limit or will have them exceed their balance, etc.), zero. At this point the bank machine may close the communication with the bank.

The bank machine will then send a request to security to do a check using the transaction identification, *tid*, supplied from the bank. The security check will determine, for example, whether the card has been stolen, or whether it has been used within the last few hours half-way around the world, etc. The bank machine then receives a response, *srp*, from security indicating that the card is okay, Accept, or is not, Deny. If the reply from security does not indicate any problem then close communication with security and provide the user with the amount *y*. If the reply from security indicates a problem then the machine will hold on to the user's card and not provide any amount of money. The program will then terminate.

We present the program using the syntax which is developed in Section 3.2. The program has type

$$\begin{aligned} \mathrm{usr} : \mathrm{Request} \circ (\mathrm{Response} \bullet \bot) \Vdash\ & \mathrm{bnk} : \mathrm{Request} \circ (\mathrm{BResponse} \bullet \bot), \\ & \mathrm{sec} : \mathrm{TransID} \circ (\mathrm{SResponse} \bullet \bot) \end{aligned}$$

and thus involves three channels labelled usr, bnk, and sec. The typing is given as

- type Request = PIN $*$ Integer
- type BResponse = TransID $*$ Integer
- data Response = Dollar Integer | TakeCard
- data SResponse = Accept | Deny

and the program is:

```
get usr (pin, x) ⇒
    put bnk (pin, x)
    get bnk (tid, y) ⇒
        close bnk
        put sec tid
        get sec srp ⇒
            case srp of
                | Accept → close sec
                           put usr (Dollar y)
                           end usr
                | Deny   → close sec
                           put usr TakeCard
                           end usr
```

The point of the example is that it shows how values received from other processes can not only affect the values subsequently passed but also the evolution of the communications of the whole process. From a proof theoretic perspective this means that the coproduct structure must be shared between the value level (the messages) and the communication level (message passing). This significantly affects the design of the logic.

It may seem to the reader that we have made a rather esoteric choice of logic for the messages. The choice is, in fact, minimal in order to illustrate the interaction between the levels. Intuitively, the reader should view messages as values of a sequential programming language (as in the above example). However, in this paper, we have not committed ourselves to a particular semantics for that sequential world. Thus, these values might be from a cartesian closed category or, equally, they could be values produced by a partial recursive function (so embody the possibility of non-termination). In this latter case, while coproducts are present, products (in the usual categorical sense) are not present nor is the setting closed. However, notably, it is an example of the minimal structure we present.

Note that in the above example a rather simple use of the unit $\bot$ is made. Those familiar with the coherence issues surrounding linearly distributive and $*$-autonomous categories will be aware that the presence of units adds significantly to the complexity of determining equality of maps. Thus, it may seem sensible to avoid these units altogether in a programming system. However, units have a crucial role as it is their behavior which allows the proper opening and closing of channels (as was seen in the above example).

The starting point of our exposition is a description of a logic for the messages and a term notation to express the proofs of this logic. The term notation for the proofs is essentially the term logic which Barry Jay developed for monoidal categories [18]. This logic does not reflect the obvious symmetry of monoidal categories obtained by reversing maps and, perhaps for that reason, it did not resonate well with work in monoidal categories. Here as we wish to contrast the one-sidedness (multicategorical nature) of message logic with the two-sidedness (polycategorical nature) of the logic for message passing, it suits our purpose well.

In more modern terms the message logic is a multicategorical logic with tensorial representation: we present the term logic from this perspective (see the work of Abramsky [1] and Mackie, Roman, and Abramsky [21] for similar systems) and add a syntax for coproducts. To present the message passing logic, we have built on the two-sided logic presented in [13]. In that paper a process reading for the two-sided terms of additive linear logic was presented. There, in order to present a two-sided notation for the proofs, we borrowed heavily from the notation of the π-calculus. This paper continues this trend by borrowing notation from the π-calculus in order to express the proofs associated with message-passing.

Having introduced the logic and a term calculus for its proofs, our next aim is to lay out the categorical semantics. We claim that this semantics lies in a "linear actegory",[1] by which we mean a linearly distributive category with a monoidal category acting on it both covariantly and contravariantly. This structure arrives with a number of coherence conditions which we have tried to lay out reasonably carefully. We believe that this description of the categorical semantics of message-passing is novel.

[1] The term *actegory* is used to describe the situation of a monoidal category "acting" on a category. They first appeared (under a different name) in the work of Bénabou as a simple example of a bicategory. B. Pareigis developed the theory of actegories (again under a different name) and showed there usefulness in the representation theory of monoids and comonoids. The word "actegory" was first suggested at the Australian Category Seminar and first appeared in print in the thesis of P. McCrudden [20] where they were used to study categories of representations of coalgebroids.

$$\text{axiom}\ \frac{}{\Phi \vdash A} \qquad \text{subs}\ \frac{\Phi \vdash A \quad \Psi_1, A, \Psi_2 \vdash B}{\Psi_1, \Phi, \Psi_2 \vdash B}$$

$$*_l\ \frac{\Phi, A, B \vdash C}{\Phi, A * B \vdash C} \qquad *_r\ \frac{\Phi \vdash A \quad \Psi \vdash B}{\Phi, \Psi \vdash A * B}$$

$$I_l\ \frac{\Phi \vdash A}{\Phi, I \vdash A} \qquad I_r\ \frac{}{\vdash I}$$

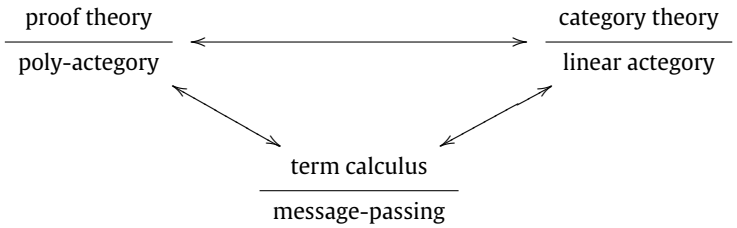
$$\text{coprod}\ \frac{\Phi, A \vdash C \quad \Phi, B \vdash C}{\Phi, A + B \vdash C} \qquad \text{inj}_l\ \frac{\Phi \vdash A}{\Phi \vdash A + B}$$

$$\mathbf{0}\ \frac{}{\Phi, \mathbf{0} \vdash A} \qquad \text{inj}_r\ \frac{\Phi \vdash B}{\Phi \vdash A + B}$$

Fig. 1. Inference rules for **Msg**.

The final aim of the paper is to connect the term calculus, the proof theory, and the categorical semantics. In order to make the paper more accessible, we begin by introducing the proof theory using a sequent-calculus presentation which is annotated to provide a term calculus. However, the sequent-calculus proof system is equivalent to a natural deduction system which in turn is a poly-actegory. In the last sections, we move rather freely between the proof theory, the poly-actegorical semantics, and the circuit representation of these systems. To one who is not familiar with these techniques these may seem like large leaps as we have not tried to provide a detailed justification of it here. These techniques are described elsewhere and originate in Lambek's work [19]. For the linear setting they are described in [10] (see also [17]), where the correctness criterion (i.e., the net condition) for the circuit representation of proofs is also discussed: this, although present in the current proof system, we barely mention.

The last sections are concerned with establishing the following three-way equivalence:



Outline of the paper. In Section 2 we introduce the logic of messages. Section 3 introduces the logic of message-passing which is built atop the logic of messages and supplies the step from the proof theory to the term calculus. Also, we introduce the cut-elimination process for the logic and so, implicitly, an operational semantics for the calculus. In Section 4 we introduce the categorical semantics. In Section 5 we show how to obtain the categorical semantics from the term calculus. In Section 6 we show how to move from the categorical semantics to the poly-actegorical semantics and back using representability. Whence, by the identification of the proof theory and the poly-actegorical semantics, we complete the tour of the triangle.

2. The logic of messages

In this section a logic for monoidal categories with coproducts is developed. The logic is presented in a Gentzen sequent style: a sequent takes the form

$$\Phi \vdash A$$

where the *antecedent* (which we will also call the *context*) of the sequent Φ is a comma separated list of formulas and the *succedent* A is a single formula. It is convenient to take the antecedent to be unordered as this allows the permutations of the formulas without having to add an explicit exchange rule:

$$\frac{\Phi_1, B, C, \Phi_2 \vdash A}{\Phi_1, C, B, \Phi_2 \vdash A}\ \text{exchange}\ .$$

The inference rules for this logic are presented in Fig. 1. Notice that the cut rule here is called "*sub*" to stand for substitution.

We will consider only the *free logic* built from a multicategory. This means that we have an arbitrary set of atoms that will be regarded as the objects of a multicategory, and an arbitrary set of axioms which will be regarded as the morphisms of a multicategory. The resulting logic will be denoted by **Msg**.

$$\frac{\text{axiom } f}{x_1 : A_1, \ldots, x_n : A_n \vdash f(x_1, \ldots, x_n) : B} \qquad \frac{\Phi \vdash f : A \quad \Psi_1, w : A, \Psi_2 \vdash g : B}{\Psi_1, \Phi, \Psi_2 \vdash (w \mapsto g) f : B}$$

$$\frac{\Phi, x : A, y : B \vdash f : C}{\Phi, (x, y) : A * B \vdash f : C} \qquad \frac{\Phi \vdash f : A \qquad \Psi \vdash g : B}{\Phi, \Psi \vdash (f, g) : A * B}$$

$$\frac{\Phi \vdash f : A}{\Phi, ( ) : I \vdash f : A} \qquad \frac{}{\vdash ( ) : I}$$

$$\frac{\Phi, x : A \vdash f : C \quad \Phi, y : B \vdash g : C}{\Phi, z : A + B \vdash \left\{ \begin{matrix} \sigma_1(x) \mapsto f \\ \sigma_2(y) \mapsto g \end{matrix} \right\} z : C} \qquad \frac{\Phi \vdash f : A}{\Phi \vdash \sigma_1(f) : A + B}$$

$$\frac{}{\Phi, z : \mathbf{0} \vdash \{ \} z : A} \qquad \frac{\Phi \vdash f : B}{\Phi \vdash \sigma_2(f) : A + B}$$

Fig. 2. Term formation rules for **Msg**.

2.1. A term calculus for **Msg**

We now introduce a term calculus for this logic. The idea is that, given a derivable sequent, to annotate the formulas on the left of the turnstile ("$\vdash$") with "patterns" made up of variables ($x, y, z, \ldots$), and the formula on the right of the turnstile with a term ($f, g, \ldots$) which together describe a derivation of the sequent.

Given a derivable annotated sequent $\Phi \vdash A$ its annotation and corresponding term are constructed inductively (top-down) from the derivation. The description is given in Fig. 2. The notation $(w \mapsto g)f$ for the substitution is to read as one would the λ-expression $(\lambda w.g)f$.

For the identity derivation on atoms $x : A \vdash A$, instead of $1_A(x)$ we will simply write x. That is, for atoms, the term formation rule is given by

$$\frac{\text{atomic } A}{x : A \vdash x : A}\,.$$

In order to avoid variable name clashes, an assumption that will be made is that whenever two or more annotated sequents are involved in a derivation (i.e., a sub, $*_r$, or coprod rule) no two will contain a variable name in common unless mentioned explicitly.

Notice that different derivations of the same sequent will be described by the same annotation. For example, notice that the derivations

$$\frac{\dfrac{\dfrac{}{x : A \vdash x : A} \quad \dfrac{\dfrac{}{\vdash ( ) : I}}{( ) : I \vdash ( ) : I}}{x : A, ( ) : I \vdash (x, ( )) : A * I}}{(x, ( )) : A * I \vdash (x, ( )) : A * I} \quad \text{and} \quad \frac{\dfrac{\dfrac{\dfrac{}{x : A \vdash x : A} \quad \dfrac{}{\vdash ( ) : I}}{x : A \vdash (x, ( )) : A * I}}{x : A, ( ) : I \vdash (x, ( )) : A * I}}{(x, ( )) : A * I \vdash (x, ( )) : A * I}$$

are both described by the annotation

$$(x, ( )) : A * I \vdash (x, ( )) : A * I$$

and are therefore implicitly identified in the term calculus.

Here are two derivations of the same sequent in which the terms describing the derivation differ.

(1)
$$\frac{\dfrac{\dfrac{}{x : A \vdash x : A} \quad \dfrac{}{y : A \vdash y : A}}{z : A + A \vdash \left\{ \begin{matrix} \sigma_1(x) \mapsto x \\ \sigma_2(y) \mapsto y \end{matrix} \right\} z : A}}{z : A + A \vdash \sigma_1 \left(\left\{ \begin{matrix} \sigma_1(x) \mapsto x \\ \sigma_2(y) \mapsto y \end{matrix} \right\} z \right) : A + B}$$

(2)
$$\frac{\dfrac{\dfrac{}{x : A \vdash x : A}}{x : A \vdash \sigma_1(x) : A + B} \quad \dfrac{\dfrac{}{y : A \vdash y : A}}{y : A \vdash \sigma_1(y) : A + B}}{z : A + A \vdash \left\{ \begin{matrix} \sigma_1(x) \mapsto \sigma_1(x) \\ \sigma_2(y) \mapsto \sigma_1(y) \end{matrix} \right\} z : A + B}$$

It will be seen in Section 2.3 that these two terms must be identified.

2.2. Cut-elimination for **Msg**

In this section the cut-("sub") elimination rewrites are described. Recall that, unless explicitly mentioned, a term may not contain a variable name in common.

Notice that the $*_l$ and I_l rules have no effect on the terms. In these cases the term is actually encoding an implicit cut-elimination step. For example, both the left-hand and right-hand derivations in the cut-elimination step

$$\dfrac{\Phi \vdash A \qquad \dfrac{\Psi_1, A, \Psi_2, B, C \vdash \mathrm{D}}{\Psi_1, A, \Psi_2, B * C \vdash D}}{\Psi_1, \Phi, \Psi_2, B * C \vdash D} \implies \dfrac{\Phi \vdash A \qquad \Psi_1, A, \Psi_2, B, C \vdash D}{\dfrac{\Psi_1, \Phi, \Psi_2, B, C \vdash D}{\Psi_1, \Phi, \Psi_2, B * C \vdash D}}$$

are represented by the same term.

In what follows the notation $f[y/x]$ will mean "in f substitute y for all occurrences of x". Also to recover the type of a term $\Phi \vdash f : A$ we denote $\mathrm{Cont}(f) = \Phi$ and $\mathrm{Out}(f) = A$.

The cut-elimination rewrites are as follows.

$$\begin{aligned}
&[\text{id-sequent}] && (w \mapsto f)\, x \implies f[x/w] \\
&[\text{sequent-id}] && (w \mapsto w)\, g \implies g \\
&[\text{sequent-}*_r] && (w \mapsto (g_1, g_2))f \implies \begin{cases} \big((w \mapsto g_1)f\,,\ g_2\big) & w \in \mathrm{Cont}(g_1) \\ \big(g_1\,,\ (w \mapsto g_2)f\big) & w \in \mathrm{Cont}(g_2) \end{cases} \\
&[\text{sequent-coprod}] && \left(w \mapsto \begin{Bmatrix} \sigma_1(x) \mapsto g_1 \\ \sigma_2(y) \mapsto g_2 \end{Bmatrix} z\right)f \implies \begin{Bmatrix} \sigma_1(x) \mapsto (w \mapsto g_1)f \\ \sigma_2(y) \mapsto (w \mapsto g_2)f \end{Bmatrix} z \\
&[\text{coprod-sequent}] && (w \mapsto g)\left(\begin{Bmatrix} \sigma_1(x) \mapsto f_1 \\ \sigma_2(y) \mapsto f_2 \end{Bmatrix} z\right) \implies \begin{Bmatrix} \sigma_1(x) \mapsto (w \mapsto g)f_1 \\ \sigma_2(y) \mapsto (w \mapsto g)f_2 \end{Bmatrix} z \\
&[\text{sequent-}\mathbf{0}] && (w \mapsto \{\}z)f \implies \{\}z \\
&[\mathbf{0}\text{-sequent}] && (w \mapsto g)(\{\}z) \implies \{\}z \\
&[\text{sequent-inj}_l] && (w \mapsto \sigma_1(g))f \implies \sigma_1((w \mapsto g)f) \\
&[\text{sequent-inj}_r] && (w \mapsto \sigma_2(g))f \implies \sigma_2((w \mapsto g)f) \\
&[*_r\text{-}*_l] && ((x, y) \mapsto g)(f_1, f_2) \implies (x \mapsto (y \mapsto g)f_2\,)f_1 \\
&[I_r\text{-}I_l] && (()\mapsto g)\,() \implies g \\
&[\text{inj}_l\text{-coprod}] && \left(z \mapsto \begin{Bmatrix} \sigma_1(x) \mapsto g_1 \\ \sigma_2(y) \mapsto g_2 \end{Bmatrix} z\right)\sigma_1(f) \implies (x \mapsto g_1)f \\
&[\text{inj}_r\text{-coprod}] && \left(z \mapsto \begin{Bmatrix} \sigma_1(x) \mapsto g_1 \\ \sigma_2(y) \mapsto g_2 \end{Bmatrix} z\right)\sigma_2(f) \implies (y \mapsto g_2)f
\end{aligned}$$

The cut-elimination procedure accounts for all the ways in which a cut can move above a compound formula which is introduced either on the left or on the right. Of course the cut-elimination procedure will get stuck on the atomic cuts (composition) in the multicategory. However, it is easy to check that, if composition terminates in the underlying multicategory, this process will terminate. Indeed, in terms with only primitive function symbols (no axioms), which only involve primitive types (no atoms), the cuts can be completely eliminated.

2.3. Equations in **Msg**

In order to ensure that the cut-elimination procedure is confluent identities (for which we use the notation " $\models\!\!=\!\!\models$ ") between cut-eliminated terms need to be introduced. Firstly, if f, g, and h are axioms, equations are needed describing the associative law and interchange law. For associativity suppose

$$\Phi \vdash f : A, \quad \Psi_1, x : A, \Psi_2 \vdash g : B, \quad \text{and} \quad \Psi_1', y : B, \Psi_2' \vdash h : B.$$

for which the identity

$$(y \mapsto h)((x \mapsto g)f) \models\!\!=\!\!\models (x \mapsto (y \mapsto h)g)f$$

describing associativity must be added. Similarly, for the interchange law suppose

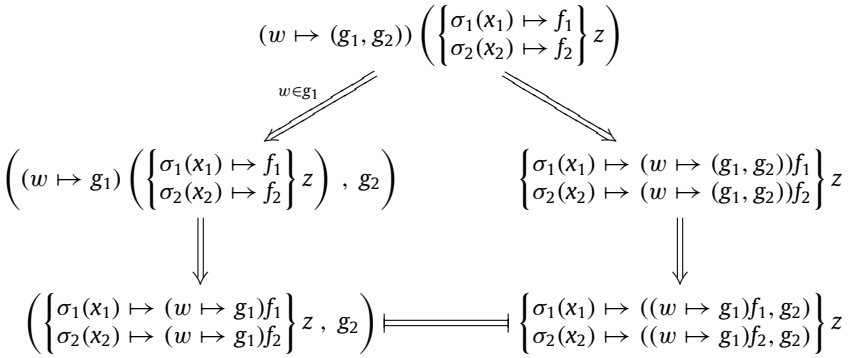
$$\Phi \vdash f : A, \quad \Phi' \vdash g : B, \quad \text{and} \quad \Psi_1, x : A, \Psi_2, y : B, \Psi_3 \vdash h : C.$$

The identity

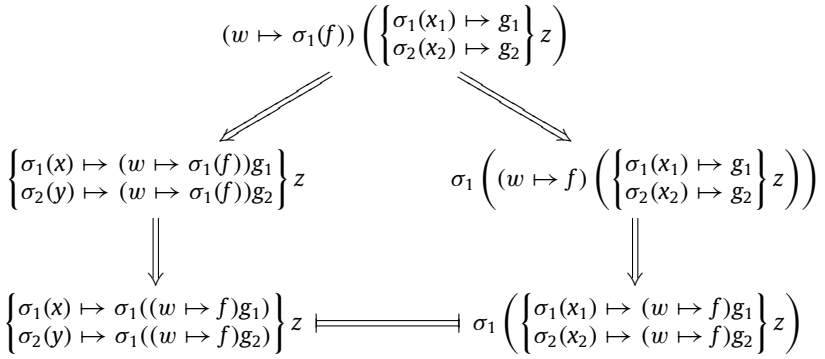
$$(y \mapsto (x \mapsto h)f)g \models\!\!=\!\!\models (x \mapsto (y \mapsto h)g)f$$

describes the interchange law.

We now move on to examining the compound terms. Here is an example which shows how such an identity arises. There are two ways to cut-eliminate the top term in the following diagram.



and this forces the identity at the bottom of the diagram. Similarly, the identification of the two terms at the end of Section 2.1 arises from the following diagram.



The list of identities which are introduced into the system in this manner is presented in Fig. 3.

Example 2.1. The cut-elimination procedure allows us to prove the distributive law. This involves proving that the composite of

$$w : A * x : (B + C) \xrightarrow[\begin{Bmatrix}\sigma_1(x_1) \mapsto \sigma_1((w, x_1))\\ \sigma_2(x_2) \mapsto \sigma_2((w, x_2))\end{Bmatrix}_x]{} A * B + A * C$$

and

$$z : ((y_1, y_2) : A * B + (z_1, z_2) : A * C) \xrightarrow[\begin{Bmatrix}\sigma_1((y_1, y_2)) \mapsto (y_1, \sigma_1(y_2))\\ \sigma_2((z_1, z_2)) \mapsto (z_1, \sigma_2(z_2))\end{Bmatrix}_z]{} A * (B + C),$$

and its reverse, are the identity. The above composite gives:

$$\left(z \mapsto \begin{Bmatrix}\sigma_1((y_1, y_2)) \mapsto (y_1, \sigma_1(y_2))\\ \sigma_2((z_1, z_2)) \mapsto (z_1, \sigma_2(z_2))\end{Bmatrix} z\right)\left(\begin{Bmatrix}\sigma_1(x_1) \mapsto \sigma_1((w, x_1))\\ \sigma_2(x_2) \mapsto \sigma_2((w, x_2))\end{Bmatrix} x\right)$$
$$\Downarrow \text{[coprod-sequent]}$$
$$\begin{Bmatrix}\sigma_1(x_1) \mapsto \left(z \mapsto \begin{Bmatrix}\sigma_1((y_1, y_2)) \mapsto (y_1, \sigma_1(y_2))\\ \sigma_2((z_1, z_2)) \mapsto (z_1, \sigma_2(z_2))\end{Bmatrix} z\right)\sigma_1((w, x_1))\\ \sigma_2(x_2) \mapsto \left(z \mapsto \begin{Bmatrix}\sigma_1((y_1, y_2)) \mapsto (y_1, \sigma_1(y_2))\\ \sigma_2((z_1, z_2)) \mapsto (z_1, \sigma_2(z_2))\end{Bmatrix} z\right)\sigma_2((w, x_2))\end{Bmatrix} x$$
$$\Downarrow \text{[inj-coprod]}$$

$$
\begin{array}{ll}
(1)\ (y \mapsto h)((x \mapsto g)f) \Longrightarrow\!\!\!\!\mid\ (x \mapsto (y \mapsto h)g)f & f,g,h \text{ atomic}\\
(2)\ (y \mapsto (x \mapsto h)f)g \Longrightarrow\!\!\!\!\mid\ (x \mapsto (y \mapsto h)g)f & f,g,h \text{ atomic}\\
(3)\ \left(\left\{\begin{array}{l}\sigma_1(x) \mapsto f_1\\ \sigma_2(y) \mapsto f_2\end{array}\right\}z, g\right) \Longrightarrow\!\!\!\!\mid\ \left\{\begin{array}{l}\sigma_1(x) \mapsto (f_1, g)\\ \sigma_2(y) \mapsto (f_2, g)\end{array}\right\}z &\\
(4)\ \left(f, \left\{\begin{array}{l}\sigma_1(x) \mapsto g_1\\ \sigma_2(y) \mapsto g_2\end{array}\right\}z\right) \Longrightarrow\!\!\!\!\mid\ \left\{\begin{array}{l}\sigma_1(x) \mapsto (f, g_1)\\ \sigma_2(y) \mapsto (f, g_2)\end{array}\right\}z &\\
(5)\ \left\{\begin{array}{l}\sigma_1(x_1) \mapsto \left\{\begin{array}{l}\sigma_1(y_1) \mapsto f_1\\ \sigma_2(y_2) \mapsto f_2\end{array}\right\}z_2\\ \sigma_2(x_2) \mapsto \left\{\begin{array}{l}\sigma_1(y_1) \mapsto g_1\\ \sigma_2(y_2) \mapsto g_2\end{array}\right\}z_2\end{array}\right\}z_1 \Longrightarrow\!\!\!\!\mid\ \left\{\begin{array}{l}\sigma_1(y_1) \mapsto \left\{\begin{array}{l}\sigma_1(x_1) \mapsto f_1\\ \sigma_2(x_2) \mapsto g_1\end{array}\right\}z_1\\ \sigma_2(y_2) \mapsto \left\{\begin{array}{l}\sigma_1(x_1) \mapsto f_2\\ \sigma_2(x_2) \mapsto g_2\end{array}\right\}z_1\end{array}\right\}z_2 &\\
(6)\ \sigma_1\left(\left\{\begin{array}{l}\sigma_1(x) \mapsto f\\ \sigma_2(y) \mapsto g\end{array}\right\}z\right) \Longrightarrow\!\!\!\!\mid\ \left\{\begin{array}{l}\sigma_1(x) \mapsto \sigma_1(f)\\ \sigma_2(y) \mapsto \sigma_1(g)\end{array}\right\}z &\\
(7)\ \sigma_2\left(\left\{\begin{array}{l}\sigma_1(x) \mapsto f\\ \sigma_2(y) \mapsto g\end{array}\right\}z\right) \Longrightarrow\!\!\!\!\mid\ \left\{\begin{array}{l}\sigma_1(x) \mapsto \sigma_2(f)\\ \sigma_2(y) \mapsto \sigma_2(g)\end{array}\right\}z &\\
(8)\ (f, \{\,\}z) \Longrightarrow\!\!\!\!\mid\ \{\,\}z &\\
(9)\ (\{\,\}z, g) \Longrightarrow\!\!\!\!\mid\ \{\,\}z &\\
(10)\ \left\{\begin{array}{l}\sigma_1(x) \mapsto \{\,\}z\\ \sigma_2(y) \mapsto \{\,\}z\end{array}\right\} \Longrightarrow\!\!\!\!\mid\ \{\,\}z &\\
(11)\ \sigma_1(\{\,\}z) \Longrightarrow\!\!\!\!\mid\ \{\,\}z &\\
(12)\ \sigma_2(\{\,\}z) \Longrightarrow\!\!\!\!\mid\ \{\,\}z &
\end{array}
$$

Fig. 3. Identities in **Msg**.

$$
\begin{array}{c}
\left\{\begin{array}{l}\sigma_1(x_1) \mapsto \big((y_1, y_2) \mapsto (y_1, \sigma_1(y_2))\big)(w, x_1)\\ \sigma_2(x_2) \mapsto \big((z_1, z_2) \mapsto (z_1, \sigma_2(z_2))\big)(w, x_2)\end{array}\right\}x\\
\Downarrow [*_r\text{-}*_l]\\
\left\{\begin{array}{l}\sigma_1(x_1) \mapsto (y_1 \mapsto (y_2 \mapsto (y_1, \sigma_1(y_2)))x_1)w\\ \sigma_2(x_2) \mapsto (z_1 \mapsto (z_2 \mapsto (z_1, \sigma_2(z_2)))x_2)w\end{array}\right\}x\\
\Downarrow \text{[id-sequent]}\\
\left\{\begin{array}{l}\sigma_1(x_1) \mapsto (y_2 \mapsto (w, \sigma_1(y_2)))x_1\\ \sigma_2(x_2) \mapsto (z_2 \mapsto (w, \sigma_2(z_2)))x_2\end{array}\right\}x\\
\Downarrow \text{[id-sequent]}\\
\left\{\begin{array}{l}\sigma_1(x_1) \mapsto (w, \sigma_1(x_1))\\ \sigma_2(x_2) \mapsto (w, \sigma_2(x_2))\end{array}\right\}x
\end{array}
$$

where this last is (one form of) the identity map of $A * (B + C)$. The other way around also works (giving the identity on $A * B + A * C$) and is left for the interested reader to familiarise themselves with this calculus.

3. The logic of message passing

This section introduces the sequent rules for the message passing logic which will be denoted by **PMsg**. As the message-passing logic is built on top of the logic of messages – which in this case we are taking to be **Msg** – the logic involves inference rules whose premises are inferences of both systems. In order to help the reader keep this straight we shall use two different entailment symbols: "$\vdash$" for the messages themselves and "$\Vdash$" for the message passing logic. For example, the inference rule $\bullet_l$ has the form

$$\frac{\dfrac{\pi}{\Phi \vdash A} \qquad \dfrac{\Pi}{\Psi \mid \Gamma, X \Vdash \Delta}}{\Phi, \Psi \mid \Gamma, A \bullet X \Vdash \Delta}$$

where π denotes a derivation in **Msg** and Π a derivation of **PMsg**.

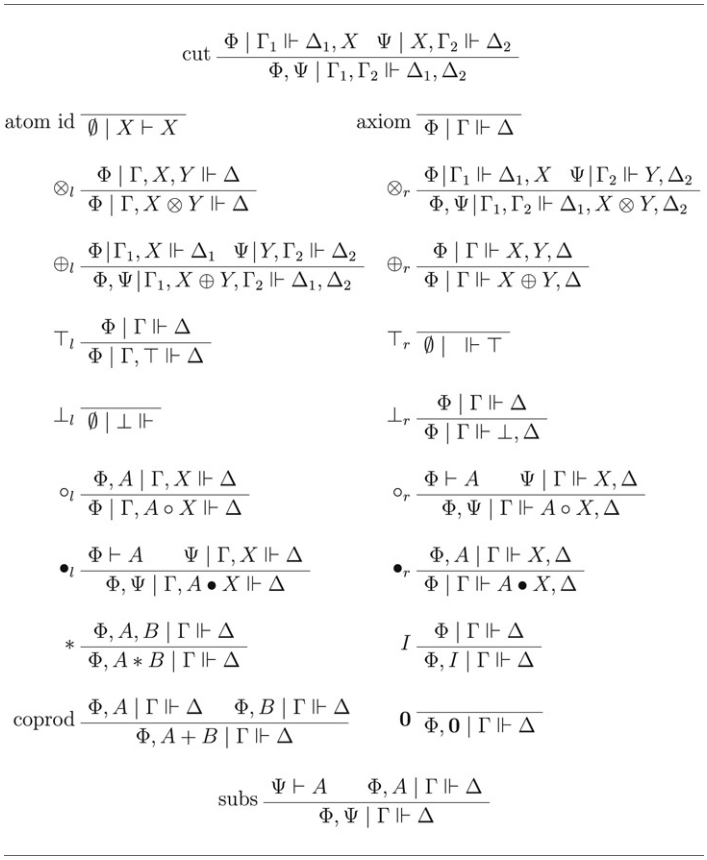


Fig. 4. Inference rules for **PMsg**.

Semantically the message passing logic builds a linear distributive category (which, when linear adjoints are present, is just a $*$-autonomous category [12]) from the underlying message logic. This linearly distributive category, as we shall see, is part of a linear actegory. The term calculus, which we construct in the next section, then becomes a very basic language for concurrent programs which can pass as values the messages provided by our message logic.

A sequent of **PMsg** has three components: the message type context Φ, its input message-passing types Γ, and its output message-passing types Δ which together define a sequent of **PMsg**:

$$\Phi \mid \Gamma \Vdash \Delta.$$

We shall treat all three components as unordered lists.

The axioms for the message-passing logic should be regarded as being maps in a poly-actegory. A poly-actegory (see Section 6) is a symmetric polycategory whose components have certain inputs from a multicategory. Intuitively one may think of such an axiom as a process between certain channels which is parameterised by certain values (from the sequential world). The usual associativity and interchange laws hold for this (multi and) polycomposition.

The inference rules for **PMsg** are presented in Fig. 4. Our convention will be to denote formulas from the message logic **Msg** using uppercase letters from the beginning of the alphabet $A, B, \ldots$ and unordered lists of these formulas using Φ and Ψ. Formulas from the message passing logic **PMsg** will be denoted using uppercase letters from the end of the alphabet $X, Y, \ldots$ and unordered lists of these formulas using Γ and Δ.

Most of the rules of this calculus are just the standard ones for two-sided multiplicative linear logic. The main novel aspects are the "action" rules $\circ_l$, $\circ_r$, $\bullet_l$, and $\bullet_r$, which allow messages to be bound to channels of interaction in the message-passing logic **PMsg**. However, also notice that the effect of a sum of messages can be derived from how they are passed in **PMsg**.

3.1. *Term calculus for* **PMsg**

The term calculus we now introduce for message-passing should be thought of as a very basic language for concurrency which permits point-to-point interactions along channels. In order to refer to the individual formulas of a sequent, which

are to be thought of as the channels through which the process embodied by the sequent interacts, we shall label them with "channel names" using lowercase Greek letters. For example

$$(x, y) : A * B, z : C \mid \alpha : W \otimes X \Vdash \beta : D \bullet Y,\ \gamma : Z.$$

In the π-calculus much emphasis is lain on how these channel names are propagated. In particular the ability to pass channel names as messages introduces scope extrusion and the necessity for channel relabelling. The calculus we present is not as free-ranging and does not allow the passing of channel names as messages. In particular, here we distinguish sharply between the world of messages and the mechanisms for message-passing. It might, therefore, be supposed that the ability to pass channel names is completely absent from this system. However, this is not the case. Although we have not chosen to concentrate on these issues (or abilities), in fact they are already present in the message-passing calculus. In the π-calculus message-passing is the *only* mechanism present and so passing of channel names has to be achieved by passing them as messages. In the current calculus, significantly, there are other mechanisms present, in particular, one can bundle channels together (using the tensor or the par) and, thus, one can pass simultaneously on a channel multiple channel names along which the receiving process can subsequently interact. The issue of scope-extrusion is actually handled in the cut-elimination procedure which, in effect, also defines the operational semantics of the system. Particularly relevant in this regard is the $\otimes_r - \otimes_l$ cut-elimination step which shows how a process can use channel names which are passed to it.

Modern process calculi are also concerned with the issue of "mobility": this means both computation carried out on mobile devices (e.g., networks that have a dynamic topology), and mobile computation (e.g., executable code that is able to move around a network). For example, the ambient calculus [11] of L. Cardelli and A. Gordon was introduced to address these issues: ambients being conceptual locations in which computation can occur. The calculus we have presented is not intended to address these issues and, indeed, is completely neutral on its "ambient" implementation. It is, of course, a pertinent issue of how to model ambient calculi categorically and proof theoretically: this may provide a useful mathematical and logical insight into these calculi.

As discussed above, the cut-elimination procedure forces the behaviour of scope in our calculus and, thus, the channel renaming which is necessary. As we are using a two-sided calculus it is possible to have completely separate name spaces for input and output channels. The "plugging together" of processes on a channel – which is a cut — binds an output channel of one process s to an input channel of another t and, thus, may be denoted by an infix syntax:

$$s\ {}_{\alpha};{}_{\beta}\ t$$

where we assume s and t have distinct output channel names and distinct input channel names. We will also allow the use of the simpler

$$s\ ;{}_{\alpha}\ t$$

where α is both an output channel bound in s and an input channel bound in t. Once a channel name is bound it can be renamed to any unused name and, indeed, this may be required to reassociate cuts:

$$(s\ {}_{\alpha};{}_{\alpha'}\ t)\ {}_{\beta};{}_{\beta'}\ u = s\ {}_{\alpha};{}_{\alpha'}\ (t\ {}_{\beta};{}_{\beta'}\ u).$$

Here this equality is only valid without renaming if $\alpha' \notin \mathrm{In}(u)$ and $\beta \notin \mathrm{Out}(s)$, although with renaming this associativity is always valid.

We now describe the term formation rules. The terms presented here make use of the self-dual nature of the logic. That is, term formation rules for dual inference rules (e.g., $\otimes_l$ and $\oplus_r$) will be identical. This agrees with the process reading of the rules when there is no distinction made between being an input channel and an output channel.

The notation "::" is used to denote the term-type membership relation, e.g., $s :: \Phi \mid \Gamma \Vdash \Delta$ means that s is a term of type $\Phi \mid \Gamma \Vdash \Delta$. The lengthy syntax will not permit us to present these rules in a table, and so we do so as a list. However, a summary of the term formation rules is provided in Fig. 5.

[cut] $$\frac{s :: \Phi \mid \Gamma_1 \Vdash \Delta_1, \alpha : X \qquad t :: \Psi \mid \beta : X, \Gamma_2 \Vdash \Delta_2}{s\ {}_{\alpha};{}_{\beta}\ t :: \Phi, \Psi \mid \Gamma_1, \Gamma_2 \Vdash \Delta_1, \Delta_2}$$

[atomic identity] $$\frac{}{\alpha =_X \beta :: \emptyset \mid \alpha : X \Vdash \beta : Y}$$

[axiom] $$\frac{s(\Phi)[\Gamma; \Delta] \text{ primitive process}}{s(x_1, \ldots, x_r)[\alpha_1, \ldots, \alpha_m; \beta_1, \ldots, \beta_n] :: x : \Phi \mid \alpha : \Gamma \Vdash \beta : \Delta}$$

where $\alpha : \Gamma$ stands for $\alpha_1 : A_1, \ldots, \alpha_m : A_m$ etc.

[$\otimes_l$ and $\oplus_r$] $$\frac{s :: \Phi \mid \Gamma, \alpha_1 : X, \alpha_2 : Y \Vdash \Delta}{\alpha\langle\alpha_1, \alpha_2\rangle \cdot s :: \Phi \mid \Gamma, \alpha : X \otimes Y \Vdash \Delta}$$ and dually

$$\frac{s :: \Phi \mid \Gamma \Vdash \alpha_1 : X, \alpha_2 : Y, \Delta}{\alpha\langle\alpha_1, \alpha_2\rangle \cdot s :: \Phi \mid \Gamma \Vdash \alpha : X \oplus Y, \Delta}$$

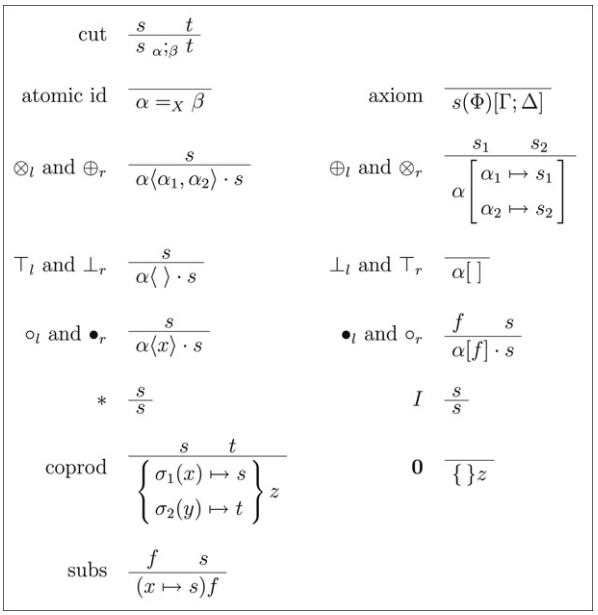


Fig. 5. Summary of the term formation rules for **PMsg**.

$$[\oplus_l \text{ and } \otimes_r] \quad \frac{s_1 :: \Phi \mid \Gamma_1, \alpha_1 : X \Vdash \Delta_1 \qquad s_2 :: \Psi \mid \alpha_2 : Y, \Gamma_2 \Vdash \Delta_2}{\alpha\begin{bmatrix}\alpha_1 \mapsto s_1\\ \alpha_2 \mapsto s_2\end{bmatrix} :: \Phi, \Psi \mid \Gamma_1, \alpha : X \oplus Y, \Gamma_2 \Vdash \Delta_1, \Delta_2}$$

and dually

$$\frac{s_1 :: \Phi \mid \Gamma_1 \Vdash \Delta_1, \alpha_1 : X \qquad s_2 :: \Psi \mid \Gamma_2 \Vdash \alpha_2 : Y, \Delta_2}{\alpha\begin{bmatrix}\alpha_1 \mapsto s_1\\ \alpha_2 \mapsto s_2\end{bmatrix} :: \Phi, \Psi \mid \Gamma_1, \Gamma_2 \Vdash \Delta_1, \alpha : X \otimes Y, \Delta_2}$$

$$[\top_l \text{ and } \bot_r] \quad \frac{s :: \Phi \mid \Gamma \Vdash \Delta}{\alpha\langle\,\rangle \cdot s :: \Phi \mid \Gamma, \alpha : \top \Vdash \Delta} \quad \text{and dually}$$

$$\frac{s :: \Phi \mid \Gamma \Vdash \Delta}{\alpha\langle\,\rangle \cdot s :: \Phi \mid \Gamma \Vdash \alpha : \bot, \Delta}$$

$$[\bot_l \text{ and } \top_r] \quad \frac{}{\alpha[\,] :: \emptyset \mid \alpha : \bot \Vdash} \quad \text{and dually}$$

$$\frac{}{\alpha[\,] :: \emptyset \mid \ \Vdash \alpha : \top}$$

$$[\circ_l \text{ and } \bullet_r] \quad \frac{s :: x : A, \Phi \mid \Gamma, \alpha : X \Vdash \Delta}{\alpha\langle x\rangle \cdot s :: \Phi \mid \Gamma, \alpha : A \circ X \Vdash \Delta} \quad \text{and dually}$$

$$\frac{s :: x : A, \Phi \mid \Gamma \Vdash \alpha : X, \Delta}{\alpha\langle x\rangle \cdot s :: \Phi \mid \Gamma \Vdash \alpha : A \bullet X, \Delta}$$

A process reading of the terms for the $[\circ_l \text{ and } \bullet_r]$ inferences may be thought of as follows: read x on channel α and bind it in the process s.

$$[\bullet_l \text{ and } \circ_r] \quad \frac{\Phi \vdash f : A \qquad s :: \Psi \mid \Gamma, \alpha : X \Vdash \Delta}{\alpha[f] \cdot s :: \Phi, \Psi \mid \Gamma, \alpha : A \bullet X \Vdash \Delta} \quad \text{and dually}$$

$$\frac{\Phi \vdash f : A \qquad s :: \Psi \mid \Gamma \Vdash \alpha : X, \Delta}{\alpha[f] \cdot s :: \Phi, \Psi \mid \Gamma \Vdash \alpha : A \circ X, \Delta}$$

A process reading of the terms for the $[\bullet_l \text{ and } \circ_r]$ inferences may be thought of as follows: output f on channel α and continue with the process s.

$$[*]\quad \frac{s :: \Phi, x : A, y : B \mid \Gamma \Vdash \Delta}{s :: \Phi, (x, y) : A * B \mid \Gamma \Vdash \Delta}$$

$$[I]\quad \frac{s :: \Phi \mid \Gamma \Vdash \Delta}{s :: \Phi, () : I \mid \Gamma \Vdash \Delta}$$

$$[\text{coprod}]\quad \frac{s :: \Phi, x : A \mid \Gamma \Vdash \Delta \qquad t :: \Phi, y : B \mid \Gamma \Vdash \Delta}{\begin{Bmatrix}\sigma_1(x) \mapsto s \\ \sigma_2(y) \mapsto t\end{Bmatrix} z :: \Phi, z : A + B \mid \Gamma \Vdash \Delta}$$

$$[\mathbf{0}]\quad \frac{}{\{\}z :: \Phi, z : \mathbf{0} \mid \Gamma \Vdash \Delta}$$

$$[\text{substitution}]\quad \frac{\Phi \vdash f : A \qquad s :: \Psi, x : A \mid \Gamma \Vdash \Delta}{(x \mapsto s)f :: \Phi, \Psi \mid \Gamma \Vdash \Delta}$$

3.2. A programming syntax term calculus

The term calculi for **PMsg** presented in the previous section is quite useful for manipulations of the logic, but it does not illustrate well the programming view of those proofs. To illustrate this relationship, and to connect with the example in the introduction, we present a programming syntax for **PMsg** similar to the syntax presented in [13].

Coproducts in programming languages are usually introduced as (non-recursive) datatypes and so it is useful to show how this may be incorporated into the message logic. A non-recursive datatype is defined as

$$\text{data } F(A_1, \ldots, A_n) = C_1 T_1 \mid \cdots \mid C_r T_r$$

where T_i is a type expression in variables $A_1, \ldots, A_n$. Note that, in contrast to the message logic in which coproducts were defined using a binary rule and a nullary rule, here we are defining coproducts indexed by an arbitrary finite set of constructors. This means in particular that this definition also captures nullary and unary coproducts.

Two rules are needed to introduce the syntax associated with this datatype:

$$\text{coprod}\quad \frac{\Phi, x_1 : T_1 \vdash f_1 : B \qquad \cdots \qquad \Phi, x_r : T_r \vdash f_r : B}{\Phi, z : F(A_1, \ldots, A_n) \vdash \text{case } z \text{ as } \begin{array}{l} \mid C_1 x_1 \to f_1 \quad : B \\ \quad \cdots \\ \mid C_r x_r \to f_r \end{array}}$$

$$\text{construct}\quad \frac{\Phi \vdash f : T_i}{\Phi \vdash C_i f : F(A_1, \ldots, A_n)}\ .$$

A convenient programming syntax for substitution is given by:

$$\text{subs}\quad \frac{\Phi \vdash f : A \qquad \Psi_1, w : A, \Psi_2 \vdash g : B}{\Psi_1, \Phi, \Psi_2 \vdash g \text{ where } w = f : B}\ .$$

A programming syntax for **PMsg** is given in Fig. 6. The only novel aspect here is the syntax for the action rules and the rules for the interaction between the two logics.

3.3. Cut elimination for **PMsg**

In this section the cut-elimination rewrites are described. Recall that unless explicitly mentioned no two subterms of a term may contain a variable or channel name in common. Also recall that the notation $s[\alpha/\beta]$ means "in s substitute α for all occurrences of β".

To recover the type of a term

$$s :: \Phi \mid \Gamma \Vdash \Delta$$

denote $\text{Cont}(s) = \Phi$, $\text{In}(s) = \Gamma$, $\text{Out}(s) = \Delta$, and $\text{Chan}(s) = \Gamma \cup \Delta$. These are respectively the *context*, *input channels*, *output channels*, and *channels* of the term s. Similarly for $\Phi \vdash f : A$ in **Msg** we have $\text{Cont}(f) = \Phi$ and $\text{Out}(f) = A$.

Since there are two types of cuts, substitution and cut, the rewrites will be split into two sections.

3.3.1. Cut rewrites

A cut in the message-passing logic has the form

$$\frac{\dfrac{\Pi}{\Phi \mid \Gamma_1 \Vdash \Delta_1, X} \qquad \dfrac{\Pi'}{\Psi \mid X, \Gamma_2 \Vdash \Delta_1}}{\Phi, \Psi \mid \Gamma_1, \Gamma_2 \Vdash \Delta_1, \Delta_1}$$

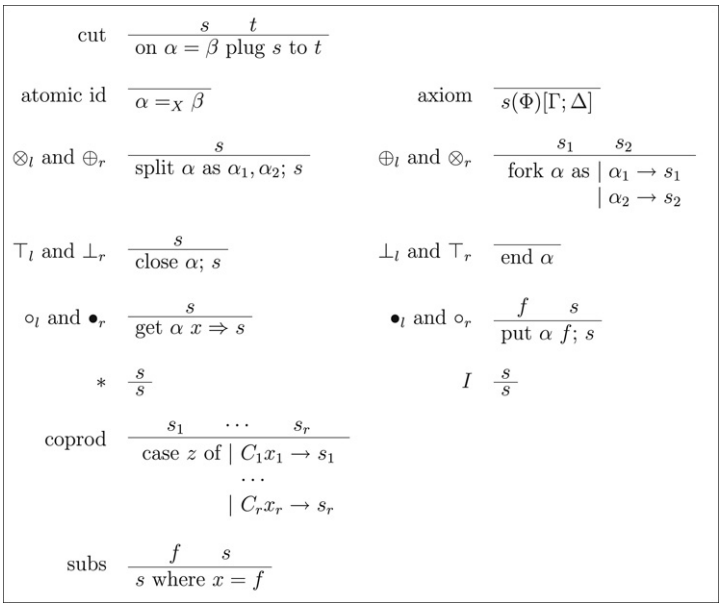


Fig. 6. Programming syntax for **PMsg**.

where both are derivations in **PMsg**. The rewrites to eliminate cuts are as follows.

$$[\text{id-sequent}] \quad \alpha =_X \beta\ {}_\beta;_\gamma\ t \implies t[\alpha/\gamma]$$

$$[\text{sequent-id}] \quad s\ {}_\gamma;_\alpha\ \alpha =_X \beta \implies s[\beta/\gamma]$$

$$[\otimes_l/\oplus_r\text{-seq}] \quad \alpha\langle\alpha_1, \alpha_2\rangle \cdot s\ {}_\beta;_\gamma\ t \implies \alpha\langle\alpha_1, \alpha_2\rangle \cdot (s\ {}_\beta;_\gamma\ t)$$

$$[\text{seq-}\otimes_l/\oplus_r] \quad s\ {}_\beta;_\gamma\ \alpha\langle\alpha_1, \alpha_2\rangle \cdot t \implies \alpha\langle\alpha_1, \alpha_2\rangle \cdot (s\ {}_\beta;_\gamma\ t)$$

$$[\oplus_l/\otimes_r\text{-seq}] \quad \alpha\begin{bmatrix}\alpha_1 \mapsto s_1\\ \alpha_2 \mapsto s_2\end{bmatrix}\ {}_\beta;_\gamma\ t \implies \begin{cases}\alpha\begin{bmatrix}\alpha_1 \mapsto s_1\ {}_\beta;_\gamma\ t\\ \alpha_2 \mapsto s_2\end{bmatrix} & \text{if } \beta \in \mathrm{Out}(s_1)\\ \alpha\begin{bmatrix}\alpha_1 \mapsto s_1\\ \alpha_2 \mapsto s_2\ {}_\beta;_\gamma\ t\end{bmatrix} & \text{if } \beta \in \mathrm{Out}(s_2)\end{cases}$$

$$[\text{seq-}\oplus_l/\otimes_r] \quad s\ {}_\beta;_\gamma\ \alpha\begin{bmatrix}\alpha_1 \mapsto t_1\\ \alpha_2 \mapsto t_2\end{bmatrix} \implies \begin{cases}\alpha\begin{bmatrix}\alpha_1 \mapsto s\ {}_\beta;_\gamma\ t_1\\ \alpha_2 \mapsto t_2\end{bmatrix} & \text{if } \gamma \in \mathrm{In}(t_1)\\ \alpha\begin{bmatrix}\alpha_1 \mapsto t_1\\ \alpha_2 \mapsto s\ {}_\beta;_\gamma\ t_2\end{bmatrix} & \text{if } \gamma \in \mathrm{In}(t_2)\end{cases}$$

$$[\top_l/\bot_r\text{-seq}] \quad \alpha\langle\,\rangle \cdot s\ {}_\beta;_\gamma\ t \implies \alpha\langle\,\rangle \cdot (s\ {}_\beta;_\gamma\ t)$$

$$[\text{seq-}\top_l/\bot_r] \quad s\ {}_\beta;_\gamma\ \alpha\langle\,\rangle \cdot t \implies \alpha\langle\,\rangle \cdot (s\ {}_\beta;_\gamma\ t)$$

$$[\circ_l/\bullet_r\text{-sequent}] \quad \alpha\langle x\rangle \cdot s\ {}_\beta;_\gamma\ t \implies \alpha\langle x\rangle \cdot (s\ {}_\beta;_\gamma\ t)$$

$$[\text{sequent-}\circ_l/\bullet_r] \quad s\ {}_\beta;_\gamma\ \alpha\langle x\rangle \cdot t \implies \alpha\langle x\rangle \cdot (s\ {}_\beta;_\gamma\ t)$$

$$[\bullet_l/\circ_r\text{-sequent}] \quad \alpha[f] \cdot s\ {}_\beta;_\gamma\ t \implies \alpha[f] \cdot (s\ {}_\beta;_\gamma\ t)$$

$$[\text{sequent-}\bullet_l/\circ_r] \quad s\ {}_\beta;_\gamma\ \alpha[f] \cdot t \implies \alpha[f] \cdot (s\ {}_\beta;_\gamma\ t)$$

$$[\text{coprod-seq}] \quad \begin{Bmatrix}\sigma_1(x) \mapsto s_1\\ \sigma_2(y) \mapsto s_2\end{Bmatrix} z\ {}_\beta;_\gamma\ t \implies \begin{Bmatrix}\sigma_1(x) \mapsto s_1\ {}_\beta;_\gamma\ t\\ \sigma_2(y) \mapsto s_2\ {}_\beta;_\gamma\ t\end{Bmatrix} z$$

$$[\text{seq-coprod}] \quad s\ {}_\beta;_\gamma\ \begin{Bmatrix}\sigma_1(x) \mapsto t_1\\ \sigma_2(y) \mapsto t_2\end{Bmatrix} z \implies \begin{Bmatrix}\sigma_1(x) \mapsto s\ {}_\beta;_\gamma\ t_1\\ \sigma_2(y) \mapsto s\ {}_\beta;_\gamma\ t_2\end{Bmatrix} z$$

$$[\mathbf{0}\text{-sequent}] \quad \{\}z\ {}_\beta;_\gamma\ t \implies \{\}z$$

$$[\text{sequent-}\mathbf{0}] \quad s\ {}_\beta;_\gamma\ \{\}z \implies \{\}z$$

$$[\otimes_r\text{-}\otimes_l] \quad \alpha\begin{bmatrix}\alpha_1 \mapsto s_1\\ \alpha_2 \mapsto s_2\end{bmatrix}\ {}_\alpha;_\beta\ \beta\langle\beta_1, \beta_2\rangle \cdot t \implies s_1\ {}_{\alpha_1};_{\beta_1}\ (s_2\ {}_{\alpha_2};_{\beta_2}\ t)$$

$$[\oplus_r\text{-}\oplus_l]\quad \alpha\langle\alpha_1,\alpha_2\rangle\cdot s\,{}_\alpha;_\beta\ \beta\begin{bmatrix}\beta_1\mapsto t_1\\ \beta_2\mapsto t_2\end{bmatrix}\implies (s\,{}_{\alpha_1};_{\beta_1}\ t_1)\,{}_{\alpha_2};_{\beta_2}\ t_2$$

$$[\top_r\text{-}\top_l]\quad \alpha[\,]\,{}_\alpha;_\beta\ \beta\langle\,\rangle\cdot t\implies t$$

$$[\bot_r\text{-}\bot_l]\quad \alpha\langle\,\rangle\cdot s\,{}_\alpha;_\beta\ \beta[\,]\implies s$$

$$[\circ_r\text{-}\circ_l]\quad \alpha[f]\cdot s\,{}_\alpha;_\beta\ \beta\langle x\rangle\cdot t\implies s\,{}_\alpha;_\beta\ (x\mapsto t)f$$

$$[\bullet_r\text{-}\bullet_l]\quad \alpha\langle x\rangle\cdot s\,{}_\alpha;_\beta\ \beta[f]\cdot t\implies (x\mapsto s)f\,{}_\alpha;_\beta\ t$$

The last two "action" cut-elimination steps (the $[\circ_r\text{-}\circ_l]$ and $[\bullet_r\text{-}\bullet_l]$ rewrites) are a bit novel so it will be nice to see an explicit sequent cut-elimination. This is the $\circ_r\text{-}\circ_l$ cut-elimination step.

$$\dfrac{\dfrac{\dfrac{\pi}{\Phi_1\vdash f:A}\quad\dfrac{\Pi}{\Phi_2\mid\Gamma_1\Vdash X,\Delta_1}}{\Phi_1,\Phi_2\mid\Gamma_1\Vdash A\circ X,\Delta_1}\quad\dfrac{\dfrac{\Pi'}{\Psi,A\mid\Gamma_2,X\Vdash\Delta_2}}{\Psi\mid\Gamma_2,A\circ X\Vdash\Delta_2}}{\Phi_1,\Phi_2,\Psi\mid\Gamma_1,\Gamma_2\Vdash\Delta_1,\Delta_2}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\Pi}{\Phi_2\mid\Gamma_1\Vdash X,\Delta_1}\quad\dfrac{\dfrac{\pi}{\Phi_1\vdash f:A}\quad\dfrac{\Pi'}{\Psi,A\mid\Gamma_2,X\Vdash\Delta_2}}{\Phi_1,\Psi\mid\Gamma_2,X\Vdash\Delta_2}}{\Phi_1,\Phi_2,\Psi\mid\Gamma_1,\Gamma_2\Vdash\Delta_1,\Delta_2}$$

Notice that there are basically two kinds of interactions which are modelled by the cut. The first is that one or the other of the terms is not active on the channel of the cut. In this case the reduction is to pass the other process into where it is active on that channel. Notice in particular how a message gets evaluated when it is passed. The other possibility is when the cut is on a channel for which, on both sides, the type formation is the leading component of the term, i.e., both terms lead with activity on that channel (the last six rewrites above). In this case we get a reduction which breaks down the type.

3.3.2. Substitution rewrites

A substitution in the message-passing logic has the form

$$\dfrac{\dfrac{\pi}{\Psi\vdash A}\quad\dfrac{\Pi}{\Phi,A\mid\Gamma\Vdash\Delta}}{\Phi,\Psi\mid\Gamma\Vdash\Delta}$$

where π is a derivation in **Msg** and Π a derivation in **PMsg**. There are eleven substitution rewrites as follows.

$$[\text{subs-}\otimes_l/\oplus_r]\quad (w\mapsto\alpha\langle\alpha_1,\alpha_2\rangle\cdot s)f\implies\alpha\langle\alpha_1,\alpha_2\rangle\cdot(w\mapsto s)f$$

$$[\text{subs-}\oplus_l/\otimes_r]\quad \left(w\mapsto\alpha\begin{bmatrix}\alpha_1\mapsto s\\ \alpha_2\mapsto t\end{bmatrix}\right)f\implies\begin{cases}\alpha\begin{bmatrix}\alpha_1\mapsto(w\mapsto s)f\\ \alpha_2\mapsto t\end{bmatrix} & \text{if } w\in\mathrm{Cont}(s)\\ \alpha\begin{bmatrix}\alpha_1\mapsto s\\ \alpha_2\mapsto(w\mapsto t)f\end{bmatrix} & \text{if } w\in\mathrm{Cont}(t)\end{cases}$$

$$[\text{subs-}\top_l/\top_r]\quad (w\mapsto\alpha\langle\,\rangle\cdot s)f\implies\alpha\langle\,\rangle\cdot(w\mapsto s)f$$

$$[\text{subs-}\circ_l/\bullet_r]\quad (w\mapsto\alpha\langle x\rangle\cdot s)f\implies\alpha\langle x\rangle\cdot(w\mapsto s)f$$

$$[\text{subs-}\bullet_l/\circ_r]\quad (w\mapsto\alpha[g]\cdot s)f\implies\begin{cases}\alpha[g]\cdot(w\mapsto s)f & \text{if } w\in\mathrm{Cont}(s)\\ \alpha[(w\mapsto g)f]\cdot s & \text{if } w\in\mathrm{Cont}(g)\end{cases}$$

$$[\text{subs-coprod}]\quad \left(w\mapsto\begin{Bmatrix}\sigma_1(x)\mapsto s\\ \sigma_2(y)\mapsto t\end{Bmatrix}z\right)f\implies\begin{Bmatrix}\sigma_1(x)\mapsto(w\mapsto s)f\\ \sigma_2(y)\mapsto(w\mapsto t)f\end{Bmatrix}z$$

$$[\text{subs-}\mathbf{0}]\quad (w\mapsto\{\,\}z)f\implies\{\,\}z$$

$$[*_r\text{-}*]\quad ((x,y)\mapsto s)(f,g)\implies(x\mapsto(y\mapsto s)g)f$$

$$[I_r\text{-}I]\quad (()\mapsto s)()\implies s$$

$$[\text{inj}_l\text{-coprod}]\quad \left(z\mapsto\begin{Bmatrix}\sigma_1(x)\mapsto s\\ \sigma_2(y)\mapsto t\end{Bmatrix}z\right)\sigma_1(f)\implies(x\mapsto s)f$$

$$[\text{inj}_r\text{-coprod}]\quad \left(z\mapsto\begin{Bmatrix}\sigma_1(x)\mapsto s\\ \sigma_2(y)\mapsto t\end{Bmatrix}z\right)\sigma_2(f)\implies(y\mapsto t)f$$

Presented as a derivation the [inj$_l$-coprod] rewrite is:

$$\dfrac{\dfrac{\dfrac{\pi}{\Phi \vdash f : X}}{\Phi \vdash \sigma_1(f) : X + Y} \qquad \dfrac{\dfrac{\Pi}{x : X, \Psi \mid \Gamma \Vdash_s \Delta} \qquad \dfrac{\Pi'}{y : Y, \Psi \mid \Gamma \Vdash_t \Delta}}{z : X + Y, \Psi \mid \Gamma \Vdash \Delta}}{\Phi, \Psi \mid \Gamma \Vdash \Delta}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\pi}{\Phi \vdash f : X} \qquad \dfrac{\Pi}{x : X, \Psi \mid \Gamma \Vdash_s \Delta}}{\Phi, \Psi \mid \Gamma \Vdash \Delta}\,.$$

3.4. Equations in **PMsg**

As in **Msg**, in order that the cut-elimination procedure is confluent identities between the cut-eliminated terms must also be added here. Slightly unusual is the fact that this system has two distinct cut rules. So that one is not given preference over the other, in addition to the usual rewrites (which do not involve either of the cut rules), an identity is required which allows the interchange of these two cut rules:

$$\dfrac{\dfrac{\Pi}{\Phi_2 \mid \Gamma_1 \Vdash X, \Delta_1} \qquad \dfrac{\dfrac{\pi}{\Phi_1 \vdash A} \qquad \dfrac{\Pi'}{\Psi, A \mid \Gamma_2, X \Vdash \Delta_2}}{\Phi_1, \Psi \mid \Gamma_2, X \Vdash \Delta_2}}{\Phi_1, \Phi_2, \Psi \mid \Gamma_1, \Gamma_2 \Vdash \Delta_1, \Delta_2}$$

$$\Updownarrow$$

$$\dfrac{\dfrac{\pi}{\Phi_1 \vdash A} \qquad \dfrac{\dfrac{\Pi}{\Phi_2 \mid \Gamma_1 \Vdash X, \Delta_1} \qquad \dfrac{\Pi'}{\Psi, A \mid \Gamma_2, X \Vdash \Delta_2}}{\Phi_2, \Psi, A \mid \Gamma_1, \Gamma_2 \Vdash \Delta_1, \Delta_2}}{\Phi_1, \Phi_2, \Psi \mid \Gamma_1, \Gamma_2 \Vdash \Delta_1, \Delta_2}$$

This is called the [cut-subs] interchange below.

In presenting the identities we shall always assume that both sides make sense. For example, the equation

$$\alpha\langle x\rangle \cdot \beta[f] \cdot s \;\Longmapsto\; \beta[f] \cdot \alpha\langle x\rangle \cdot s$$

only makes sense if x does not occur in $\mathrm{Cont}(f)$.

Many of the identities below simply express that actions on one channel should be independent – as far as is possible given the bindings – from actions on another. Thus, although there may seem to be a lot of equations their genus is quite simple.

- The two cut rewrites.

  [subs-cut] $(x \mapsto s)f\ {}_\beta;_\gamma\ t \;\Longmapsto\; (x \mapsto s\ {}_\beta;_\gamma\ t)f$

  [cut-subs] $s\ {}_\beta;_\gamma\ (x \mapsto t)f \;\Longmapsto\; (x \mapsto s\ {}_\beta;_\gamma\ t)f$

- Rewrites involving $\otimes_l$ or $\oplus_r$.

  $[\otimes_l/\oplus_r\text{-}\otimes_l/\oplus_r]$ $\alpha\langle\alpha_1, \alpha_2\rangle \cdot \beta\langle\beta_1, \beta_2\rangle \cdot s \;\Longmapsto\; \beta\langle\beta_1, \beta_2\rangle \cdot \alpha\langle\alpha_1, \alpha_2\rangle \cdot s$

  $[\otimes_l/\oplus_r\text{-}\oplus_l/\otimes_r]$

$$\alpha\langle\alpha_1, \alpha_2\rangle \cdot \begin{bmatrix} \beta_1 \mapsto s \\ \beta_2 \mapsto t \end{bmatrix} \;\Longmapsto\; \begin{cases} \beta\begin{bmatrix} \beta_1 \mapsto \alpha\langle\alpha_1, \alpha_2\rangle \cdot s \\ \beta_2 \mapsto t \end{bmatrix} & \text{if } \alpha_1, \alpha_2 \in \mathrm{Chan}(s) \\ \beta\begin{bmatrix} \beta_1 \mapsto s \\ \beta_2 \mapsto \alpha\langle\alpha_1, \alpha_2\rangle \cdot t \end{bmatrix} & \text{if } \alpha_1, \alpha_2 \in \mathrm{Chan}(t) \end{cases}$$

  $[\otimes_l/\oplus_r\text{-}\top_l/\bot_r]$ $\alpha\langle\alpha_1, \alpha_2\rangle \cdot \beta\langle\,\rangle \cdot s \;\Longmapsto\; \beta\langle\,\rangle \cdot \alpha\langle\alpha_1, \alpha_2\rangle \cdot s$

  $[\otimes_l/\oplus_r\text{-}\circ_l/\bullet_r]$ $\alpha\langle\alpha_1, \alpha_2\rangle \cdot \beta\langle x\rangle \cdot f \;\Longmapsto\; \beta\langle x\rangle \cdot \alpha\langle\alpha_1, \alpha_2\rangle \cdot f$

  $[\otimes_l/\oplus_r\text{-}\bullet_l/\circ_r]$ $\alpha\langle\alpha_1, \alpha_2\rangle \cdot \beta[f] \cdot s \;\Longmapsto\; \beta[f] \cdot \alpha\langle\alpha_1, \alpha_2\rangle \cdot s$

  $[\otimes_l/\oplus_r\text{-coprod}]$ $\alpha\langle\alpha_1, \alpha_2\rangle \cdot \begin{Bmatrix} \sigma_1(x) \mapsto s \\ \sigma_2(y) \mapsto t \end{Bmatrix} z \;\Longmapsto\; \begin{Bmatrix} \sigma_1(x) \mapsto \alpha\langle\alpha_1, \alpha_2\rangle \cdot s \\ \sigma_2(y) \mapsto \alpha\langle\alpha_1, \alpha_2\rangle \cdot t \end{Bmatrix} z$

  $[\otimes_l/\oplus_r\text{-}\mathbf{0}]$ $\alpha\langle\alpha_1, \alpha_2\rangle \cdot \{\}z \;\Longmapsto\; \{\}z$

- Rewrites involving $\oplus_l$ or $\otimes_r$. We only describe an interchange occurring on the left-hand (the α_1) branch. Similar rewrites are needed for the right-hand (the α_2) branch, but are omitted here as they are easy to infer.

$$[\oplus_l/\otimes_r\text{-}\oplus_l/\otimes_r] \quad \alpha\begin{bmatrix}\alpha_1 \mapsto \beta\begin{bmatrix}\beta_1 \mapsto s_1\\ \beta_2 \mapsto s_2\end{bmatrix}\\ \alpha_2 \mapsto t\end{bmatrix} \Longrightarrow \beta\begin{bmatrix}\beta_1 \mapsto \alpha\begin{bmatrix}\alpha_1 \mapsto s_1\\ \alpha_2 \mapsto t\end{bmatrix}\\ \beta_2 \mapsto s_2\end{bmatrix}$$

$$[\oplus_l/\otimes_r\text{-}\top_l/\bot_r] \quad \alpha\begin{bmatrix}\alpha_1 \mapsto \beta\langle\,\rangle \cdot s\\ \alpha_2 \mapsto t\end{bmatrix} \Longrightarrow \beta\langle\,\rangle \cdot \alpha\begin{bmatrix}\alpha_1 \mapsto s\\ \alpha_2 \mapsto t\end{bmatrix}$$

$$[\oplus_l/\otimes_r\text{-}\circ_l/\bullet_r] \quad \alpha\begin{bmatrix}\alpha_1 \mapsto \beta\langle x\rangle \cdot s\\ \alpha_2 \mapsto t\end{bmatrix} \Longrightarrow \beta\langle x\rangle \cdot \alpha\begin{bmatrix}\alpha_1 \mapsto s\\ \alpha_2 \mapsto t\end{bmatrix}$$

$$[\oplus_l/\otimes_r\text{-}\bullet_l/\circ_r] \quad \alpha\begin{bmatrix}\alpha_1 \mapsto \beta[f] \cdot s\\ \alpha_2 \mapsto t\end{bmatrix} \Longrightarrow \beta[f] \cdot \alpha\begin{bmatrix}\alpha_1 \mapsto s\\ \alpha_2 \mapsto t\end{bmatrix}$$

$$[\oplus_l/\otimes_r\text{-coprod}] \quad \alpha\begin{bmatrix}\alpha_1 \mapsto \begin{Bmatrix}\sigma_1(x) \mapsto s_1\\ \sigma_2(y) \mapsto s_2\end{Bmatrix} z\\ \alpha_2 \mapsto t\end{bmatrix} \Longrightarrow \begin{Bmatrix}\sigma_1(x) \mapsto \alpha\begin{bmatrix}\alpha_1 \mapsto s_1\\ \alpha_2 \mapsto t\end{bmatrix}\\ \sigma_2(y) \mapsto \alpha\begin{bmatrix}\alpha_1 \mapsto s_2\\ \alpha_2 \mapsto t\end{bmatrix}\end{Bmatrix} z$$

$$[\oplus_l/\otimes_r\text{-}\mathbf{0}] \quad \alpha\begin{bmatrix}\alpha_1 \mapsto \{\,\}z\\ \alpha_2 \mapsto t\end{bmatrix} \Longrightarrow \{\,\}z$$

• Rewrites involving $\top_l$ or $\bot_r$.

$$[\top_l/\bot_r\text{-}\top_l/\bot_r] \quad \alpha\langle\,\rangle \cdot \beta\langle\,\rangle \cdot s \Longrightarrow \beta\langle\,\rangle \cdot \alpha\langle\,\rangle \cdot s$$

$$[\top_l/\bot_r\text{-}\circ_l/\bullet_r] \quad \alpha\langle\,\rangle \cdot \beta\langle x\rangle \cdot s \Longrightarrow \beta\langle x\rangle \cdot \alpha\langle\,\rangle \cdot s$$

$$[\top_l/\bot_r\text{-}\bullet_l/\circ_r] \quad \alpha\langle\,\rangle \cdot \beta[f] \cdot s \Longrightarrow \beta[f] \cdot \alpha\langle\,\rangle \cdot s$$

$$[\top_l/\bot_r\text{-coprod}] \quad \alpha\langle\,\rangle \cdot \beta\begin{Bmatrix}\sigma_1(x) \mapsto s\\ \sigma_2(y) \mapsto t\end{Bmatrix} z \Longrightarrow \begin{Bmatrix}\sigma_1(x) \mapsto \beta\langle\,\rangle \cdot s\\ \sigma_2(y) \mapsto \beta\langle\,\rangle \cdot t\end{Bmatrix} z$$

$$[\top_l/\bot_r\text{-}\mathbf{0}] \quad \alpha\langle\,\rangle \cdot \beta\{\,\} \cdot s \Longrightarrow \beta\{\,\}$$

• Rewrites involving $\circ_l$ or $\bullet_r$.

$$[\circ_l/\bullet_r\text{-}\circ_l/\bullet_r] \quad \alpha\langle x\rangle \cdot \beta\langle y\rangle \cdot s \Longrightarrow \beta\langle y\rangle \cdot \alpha\langle x\rangle \cdot s$$

$$[\circ_l/\bullet_r\text{-}\bullet_l/\circ_r] \quad \alpha\langle x\rangle \cdot \beta[f] \cdot s \Longrightarrow \beta[f] \cdot \alpha\langle x\rangle \cdot s$$

$$[\circ_l/\bullet_r\text{-coprod}] \quad \alpha\langle x\rangle \cdot \begin{Bmatrix}\sigma_1(y_1) \mapsto s\\ \sigma_2(y_2) \mapsto t\end{Bmatrix} z \Longrightarrow \begin{Bmatrix}\sigma_1(y_1) \mapsto \alpha\langle x\rangle \cdot s\\ \sigma_2(y_2) \mapsto \alpha\langle x\rangle \cdot t\end{Bmatrix} z$$

$$[\circ_l/\bullet_r\text{-}\mathbf{0}] \quad \alpha\langle x\rangle \cdot \{\,\}z \Longrightarrow \{\,\}z$$

• Rewrites involving $\bullet_l$ or $\circ_r$. In this case we must also investigate the message terms which are passed by the $\bullet_l$ and $\circ_r$ rules as they may also lead to interchanges with the coproduct or **0** rules. Thus, the two distinct interchanges for each of the coproduct and **0** rule.

$$[\bullet_l/\circ_r\text{-}\bullet_l/\circ_r] \quad \alpha[f] \cdot \beta[g] \cdot s \Longrightarrow \beta[g] \cdot \alpha[f] \cdot s$$

$$[\bullet_l/\circ_r\text{-coprod}] \quad \alpha[f] \cdot \begin{Bmatrix}\sigma_1(x) \mapsto s\\ \sigma_2(y) \mapsto t\end{Bmatrix} z \Longrightarrow \begin{Bmatrix}\sigma_1(x) \mapsto \alpha[f] \cdot s\\ \sigma_2(y) \mapsto \alpha[f] \cdot t\end{Bmatrix} z$$

$$[\bullet_l/\circ_r\text{-coprod}] \quad \alpha\left[\begin{Bmatrix}\sigma_1(x) \mapsto f\\ \sigma_2(y) \mapsto g\end{Bmatrix} z\right] \cdot s \Longrightarrow \begin{Bmatrix}\sigma_1(x) \mapsto \alpha[f] \cdot s\\ \sigma_2(y) \mapsto \alpha[g] \cdot s\end{Bmatrix} z$$

$$[\bullet_l/\circ_r\text{-}\mathbf{0}] \quad \alpha[f] \cdot \{\,\}z \Longrightarrow \{\,\}z$$

$$[\bullet_l/\circ_r\text{-}\mathbf{0}] \quad \alpha[\{\,\}z] \cdot s \Longrightarrow \{\,\}z$$

• Rewrites involving the coproduct.

[coprod-coprod]

$$\begin{Bmatrix}\sigma_1(x_1) \mapsto \begin{Bmatrix}\sigma_1(y_1) \mapsto s_1\\ \sigma_2(y_2) \mapsto t_1\end{Bmatrix} w\\ \sigma_2(x_2) \mapsto \begin{Bmatrix}\sigma_1(y_1) \mapsto s_2\\ \sigma_2(y_2) \mapsto t_2\end{Bmatrix} w\end{Bmatrix} z \Longrightarrow \begin{Bmatrix}\sigma_1(y_1) \mapsto \begin{Bmatrix}\sigma_1(x_1) \mapsto s_1\\ \sigma_2(x_2) \mapsto s_2\end{Bmatrix} z\\ \sigma_2(y_2) \mapsto \begin{Bmatrix}\sigma_1(x_1) \mapsto t_1\\ \sigma_2(x_2) \mapsto t_2\end{Bmatrix} z\end{Bmatrix} w$$

$$[\text{coprod-}\mathbf{0}] \quad \begin{Bmatrix}\sigma_1(x) \mapsto \{\,\}z\\ \sigma_2(y) \mapsto \{\,\}z\end{Bmatrix} w \Longrightarrow \{\,\}z$$

We shall demonstrate how these identities are used in Section 5 where we relate this term logic to the categorical semantics.

4. Linear actegories

Given a (symmetric) monoidal category $\mathscr{A}$ we introduce the notion of a linear $\mathscr{A}$-actegory, which is a linear distributive category $\mathscr{X}$ equipped with two functors

$$\circ : \mathscr{A} \times \mathscr{X} \longrightarrow \mathscr{X} \qquad \text{and} \qquad \bullet : \mathscr{A}^{\mathrm{op}} \times \mathscr{X} \longrightarrow \mathscr{X},$$

the "actions" of $\mathscr{A}$ on $\mathscr{X}$. These must satisfy a number of coherence conditions which are described below.

Our aim, in the following section, is to show that these form the categorical semantics for the proof theory of **PMsg**.

4.1. Linear distributive categories

A *linear distributive category* is a category $\mathscr{X}$ equipped with a "tensor" $\otimes : \mathscr{X} \times \mathscr{X} \longrightarrow \mathscr{X}$ with unit $\top$ and coherent natural isomorphisms

$$a_{\otimes} : (X \otimes Y) \otimes Z \longrightarrow X \otimes (Y \otimes Z), \quad l_{\otimes} : \top \otimes X \longrightarrow X, \quad r_{\otimes} : X \otimes \top \longrightarrow X,$$

and a "par" $\oplus : \mathscr{X} \times \mathscr{X} \longrightarrow \mathscr{X}$ with unit $\bot$ and coherent natural isomorphism

$$a_{\oplus} : X \oplus (Y \oplus Z) \longrightarrow (X \oplus Y) \oplus Z, \quad l_{\oplus} : X \longrightarrow \bot \oplus X, \quad r_{\oplus} : X \longrightarrow X \oplus \bot$$

(the odd choice of direction is used to maximise the symmetry below), and two linear distributions

$$d^{\otimes}_{\oplus} : X \otimes (Y \oplus Z) \longrightarrow (X \otimes Y) \oplus Z \quad \text{and} \quad d^{\oplus}_{\otimes} : (Y \oplus Z) \otimes X \longrightarrow Y \oplus (Z \otimes X)$$

relating the two structures. This data must satisfy several coherence conditions (see [15]).

If both the tensor and the par are symmetric (with $c_{\otimes}$ and $c_{\oplus}$) and several other coherence conditions are satisfied (again see [15]) then it is called a *symmetric* linear distributive category. In this case there are two induced "permuting" linear distributions

$$d^{\otimes'}_{\oplus} : X \otimes (Y \oplus Z) \longrightarrow Y \oplus (X \otimes Z) \quad \text{and} \quad d^{\oplus'}_{\otimes} : (Y \oplus Z) \otimes X \longrightarrow (Y \otimes X) \oplus Z.$$

Example 4.1. (1) Any distributive lattice is a linear distributive category with the objects being the elements and the maps being comparisons: $\otimes$ is the meet and $\oplus$ is the join.

(2) Any monoidal category gives rise to a "compact" (i.e., $\otimes = \oplus$) linear distributive category. When both $\otimes$ and $\oplus$ are interpreted by the same tensor the linear distribution becomes associativity.

(3) Any $*$-autonomous category is a linearly distributive with $A \oplus B = \{\}^*(B^* \otimes A^*)$.

(4) The category of sets with $\otimes = \times$ and $\oplus$ given as follows (due to Jürgen Koslowski):

$$A \oplus B = \begin{cases} B & A = \emptyset \\ A & B = \emptyset \\ 1 & \text{otherwise.} \end{cases}$$

This is an example of a non-compact, non-posetal, non-$*$-autonomous linear distributive category.

4.2. Linear actegories

Let $\mathscr{A} = (\mathscr{A}, *, I, a_*, l_*, r_*, c_*)$ be a symmetric monoidal category. A *(symmetric) linear $\mathscr{A}$-actegory* consists of the following data.

- A symmetric linear distributive category $\mathscr{X}$ (as above),
- Functors

  $$\circ : \mathscr{A} \times \mathscr{X} \longrightarrow \mathscr{X} \quad \text{and} \quad \bullet : \mathscr{A}^{\mathrm{op}} \times \mathscr{X} \longrightarrow \mathscr{X},$$

  such that $\circ$ is the left parameterised left adjoint of $\bullet$, i.e., for all $A \in \mathscr{A}$, $A \circ - \dashv A \bullet -$. The unit and counit of this adjunction (natural in $A \in \mathscr{A}$ and $X \in \mathscr{X}$) are denoted respectively by

  $$n_{A,X} : X \longrightarrow A \bullet (A \circ X) \quad \text{and} \quad e_{A,X} : A \circ (A \bullet X) \longrightarrow X.$$

- For all $A, B \in \mathscr{A}$ and $X, Y \in \mathscr{X}$ natural isomorphisms in $\mathscr{X}$

  $$u_{\circ} : I \circ X \longrightarrow X,$$
  $$u_{\bullet} : X \longrightarrow I \bullet X,$$
  $$a^*_{\circ} : (A * B) \circ X \longrightarrow A \circ (B \circ X),$$
  $$a^*_{\bullet} : A \bullet (B \bullet X) \longrightarrow (A * B) \bullet X,$$
  $$a^{\circ}_{\otimes} : A \circ (X \otimes Y) \longrightarrow (A \circ X) \otimes Y,$$
  $$a^{\bullet}_{\oplus} : (A \bullet X) \oplus Y \longrightarrow A \bullet (X \oplus Y).$$

- For all $A, B \in \mathscr{A}$ and $X, Y \in \mathscr{X}$ natural morphisms in $\mathscr{X}$

$$d^{\circ}_{\oplus} : A \circ (X \oplus Y) \longrightarrow (A \circ X) \oplus Y,$$
$$d^{\bullet}_{\otimes} : (A \bullet X) \otimes Y \longrightarrow A \bullet (X \otimes Y),$$
$$d^{\circ}_{\bullet} : A \circ (B \bullet X) \longrightarrow B \bullet (A \circ X).$$

The symmetries of $*$, $\otimes$, and $\oplus$ induce the following permuting morphisms (or isomorphisms):

$$a^{*'}_{\circ} : (A * B) \circ X \longrightarrow B \circ (A \circ X),$$
$$a^{*'}_{\bullet} : B \bullet (A \bullet X) \longrightarrow (A * B) \bullet X,$$
$$a^{\circ}_{\otimes'} : A \circ (X \otimes Y) \longrightarrow X \otimes (A \circ Y),$$
$$a^{\bullet}_{\oplus'} : X \oplus (A \bullet Y) \longrightarrow A \bullet (X \oplus Y),$$
$$d^{\circ}_{\oplus'} : A \circ (X \oplus Y) \longrightarrow X \oplus (A \circ Y),$$
$$d^{\bullet}_{\otimes'} : X \otimes (A \bullet Y) \longrightarrow A \bullet (X \otimes Y).$$

This data must satisfy several coherence conditions which we shall discuss shortly. Firstly we try to give some intuition behind the notation that has been chosen. The "a" maps are (invertible) associativity isomorphisms and the "d" maps are (non-invertible) linear distributions. The direction of the maps have been chosen (when there is a choice) to maximise the amount of symmetry and so that the $\circ$ is pushed in a bracket and the $\bullet$ is pulled out of a bracket. This choice of notation may allow us to leave off the subscripts and let the types disambiguate the maps (which is not however done here).

The symmetries of this data are as follows:

[op′] Reverse the arrows and swap $\otimes$ and $\oplus$, $\top$ and $\bot$, and $\circ$ and $\bullet$. This gives the following assignment of generating maps

$$\begin{array}{llll} a_{\otimes} \longleftrightarrow a_{\oplus} & l_{\otimes} \longleftrightarrow l_{\oplus} & r_{\otimes} \longleftrightarrow r_{\oplus} & c_{\otimes} \longleftrightarrow c^{-1}_{\oplus} \\ u_{\circ} \longleftrightarrow u_{\bullet} & a^{*}_{\circ} \longleftrightarrow a^{*}_{\bullet} & a^{\circ}_{\otimes} \longleftrightarrow a^{\bullet}_{\oplus} & \\ d^{\otimes}_{\oplus} \longleftrightarrow d^{\oplus}_{\otimes} & d^{\circ}_{\oplus} \longleftrightarrow d^{\bullet}_{\otimes} & d^{\circ}_{\bullet} \longleftrightarrow d^{\circ}_{\bullet} & \\ n \longleftrightarrow e & & & \end{array}$$

with the remainder of the morphisms unchanged.

[$*'$] Reverse the $*$ (i.e., $A *' B = B * A$); this assigns

$$a_{*} \longleftrightarrow a^{-1}_{*} \qquad l_{*} \longleftrightarrow r_{*} \qquad c_{*} \longleftrightarrow c^{-1}_{*} \qquad a^{*}_{\circ} \longleftrightarrow a^{*'}_{\circ} \qquad a^{*}_{\bullet} \longleftrightarrow a^{*'}_{\bullet}$$

with the remainder unchanged.

[$\otimes'$] Reverse the $\otimes$; this assigns

$$\begin{array}{llll} a_{\otimes} \longleftrightarrow a^{-1}_{\otimes} & l_{\otimes} \longleftrightarrow r_{\otimes} & c_{\otimes} \longleftrightarrow c^{-1}_{\otimes} & \\ d^{\otimes}_{\oplus} \longleftrightarrow d^{\oplus'}_{\otimes} & d^{\oplus}_{\otimes} \longleftrightarrow d^{\otimes'}_{\oplus} & a^{\circ}_{\otimes} \longleftrightarrow a^{\circ}_{\otimes'} & d^{\bullet}_{\otimes} \longleftrightarrow d^{\bullet}_{\otimes'} \end{array}$$

with the remainder unchanged.

[$\oplus'$] Reverse the $\oplus$; this assigns

$$\begin{array}{llll} a_{\oplus} \longleftrightarrow a^{-1}_{\oplus} & l_{\oplus} \longleftrightarrow r_{\oplus} & c_{\oplus} \longleftrightarrow c^{-1}_{\oplus} & \\ d^{\otimes}_{\oplus} \longleftrightarrow d^{\otimes'}_{\oplus} & d^{\oplus}_{\otimes} \longleftrightarrow d^{\oplus'}_{\otimes} & a^{\bullet}_{\oplus} \longleftrightarrow a^{\bullet}_{\oplus'} & d^{\circ}_{\oplus} \longleftrightarrow d^{\circ}_{\oplus'} \end{array}$$

with the remainder unchanged.

[$*'$] Reverse the $*$; this assigns

$$a_{*} \longleftrightarrow a^{-1}_{*} \qquad l_{*} \longleftrightarrow l^{-1}_{*} \qquad r_{*} \longleftrightarrow r^{-1}_{*} \qquad c_{*} \longleftrightarrow c^{-1}_{*}$$

with the remainder unchanged.

[$\odot'$] There are four remaining symmetries obtained by reversing any combination of two or more of $*$, $\otimes$, and $\oplus$. The assignments are evident.

The notion of a linear $\mathscr{A}$-actegory is preserved by these symmetries. It is important to notice that the first symmetry is the most significant as it indicates a fundamental relationship between *different* functorial operations.

The coherence conditions for a linear $\mathscr{A}$-actegory are now described.

[Symmetries.] The two diagrams below linking the symmetries and the associativities must commute.

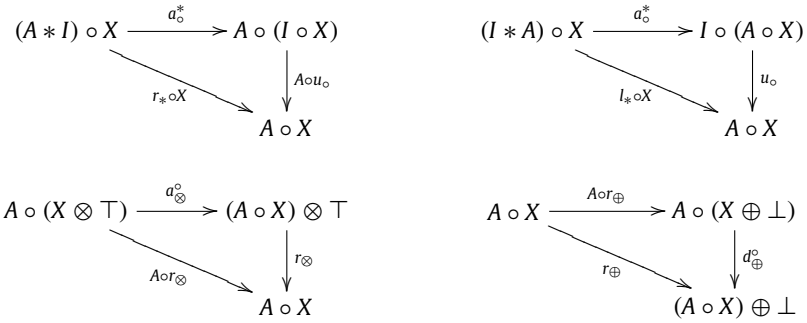
$$\begin{array}{ccc} A \circ (X \otimes Y) & \xrightarrow{a^{\circ}_{\otimes}} & (A \circ X) \otimes Y \\ \downarrow{\scriptstyle A \circ c_{\otimes}} & & \downarrow{\scriptstyle c_{\otimes}} \\ A \circ (Y \otimes X) & \xrightarrow{a^{\circ}_{\otimes'}} & Y \otimes (A \circ X) \end{array} \qquad \begin{array}{ccc} A \circ (X \oplus Y) & \xrightarrow{d^{\circ}_{\oplus}} & (A \circ X) \oplus Y \\ \downarrow{\scriptstyle A \circ c_{\oplus}} & & \downarrow{\scriptstyle c_{\oplus}} \\ A \circ (Y \oplus X) & \xrightarrow{a^{\circ}_{\oplus'}} & Y \oplus (A \circ X) \end{array}$$

These diagrams and the results of applying the symmetries to them yield the following equations:

$$a^{\circ}_{\otimes}; c_{\otimes} = (A \circ c_{\otimes}); a^{\circ}_{\otimes'} \qquad c_{\otimes}; a^{\bullet}_{\oplus} = a^{\bullet}_{\oplus'}; (A \bullet c_{\oplus}) \tag{1}$$

$$d^{\circ}_{\oplus}; c_{\oplus} = (A \circ c_{\oplus}); a^{\circ}_{\oplus'} \qquad c_{\oplus}; d^{\bullet}_{\otimes} = a^{\bullet}_{\otimes'}; (A \bullet c_{\otimes}) \tag{2}$$

[Unit and associativity.] The following diagrams linking the unit and associatively morphisms must commute.



These result in the following equations:

$$a^{*}_{\circ}; (A \circ u_{\circ}) = r_{*} \circ X \qquad (A \bullet u_{\bullet}); a^{*}_{\bullet} = r^{-1}_{*} \bullet X \tag{3}$$

$$a^{*}_{\circ}; u_{\circ} = l_{*} \circ X \qquad u_{\bullet}; a^{*}_{\bullet} = l^{-1}_{*} \bullet X \tag{4}$$

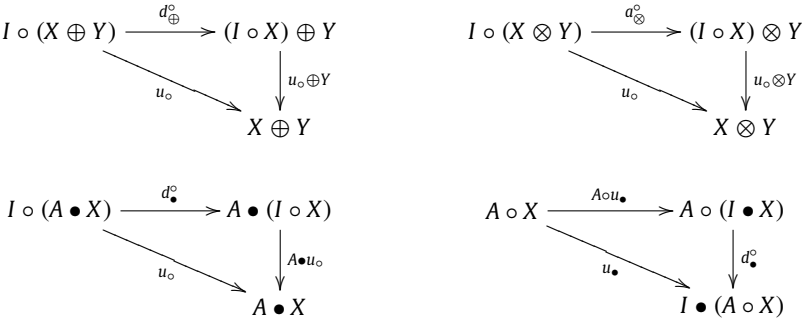
$$a^{\circ}_{\otimes}; r_{\otimes} = A \circ r_{\otimes} \qquad r_{\oplus}; a^{\bullet}_{\oplus} = A \bullet r_{\oplus} \tag{5}$$

$$a^{\circ}_{\otimes'}; l_{\otimes} = A \circ l_{\otimes} \qquad l_{\oplus}; a^{\bullet}_{\oplus'} = A \bullet l_{\oplus}$$

$$(A \circ r_{\oplus}); d^{\circ}_{\oplus} = r_{\oplus} \qquad d^{\bullet}_{\otimes}; (A \bullet r_{\otimes}) = r_{\otimes} \tag{6}$$

$$(A \circ l_{\oplus}); d^{\circ}_{\oplus'} = l_{\oplus} \qquad d^{\bullet}_{\otimes'}; (A \bullet l_{\otimes}) = l_{\otimes}$$

[Unit and distributivity.] The following diagrams linking the unit and distributivity morphisms must commute.



These result in the following equations:

$$d^{\circ}_{\oplus}; (u_{\circ} \oplus Y) = u_{\circ} \qquad (u_{\bullet} \otimes Y); d^{\bullet}_{\otimes} = u_{\bullet} \tag{7}$$

$$d^{\circ}_{\oplus'}; (Y \oplus u_{\circ}) = u_{\circ} \qquad (Y \otimes u_{\bullet}); d^{\bullet}_{\otimes'} = u_{\bullet}$$

$$a^{\circ}_{\otimes}; (u_{\circ} \otimes Y) = u_{\circ} \qquad (u_{\bullet} \oplus Y); a^{\bullet}_{\oplus} = u_{\bullet} \tag{8}$$

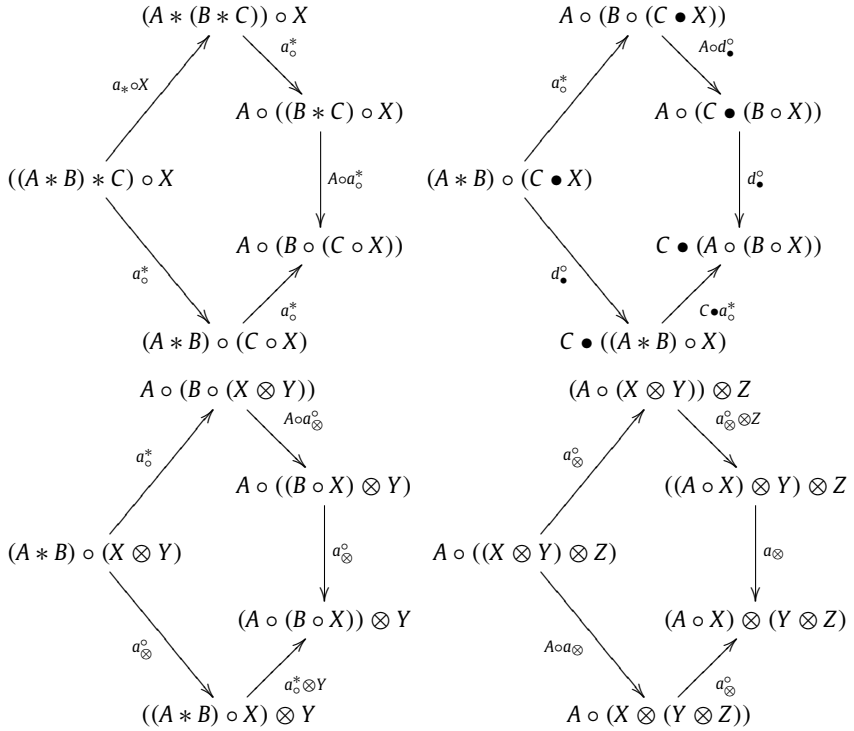
$$a^{\circ}_{\otimes'}; (Y \otimes u_{\circ}) = u_{\circ} \qquad (Y \oplus u_{\bullet}); a^{\bullet}_{\oplus'} = u_{\bullet}$$

$$d^{\circ}_{\bullet}; (A \bullet u_{\circ}) = u_{\circ} \qquad (A \circ u_{\bullet}); d^{\circ}_{\bullet} = u_{\bullet}$$

$$d^{\circ}_{\bullet}; (A \bullet u_{\circ}) = u_{\circ} \qquad (A \circ u_{\bullet}); d^{\circ}_{\bullet} = u_{\bullet} \tag{9}$$

$$A \circ u_{\bullet}; d^{\circ}_{\bullet} = u_{\bullet} \qquad d^{\circ}_{\bullet}; A \bullet u_{\circ} = u_{\circ} \tag{10}$$

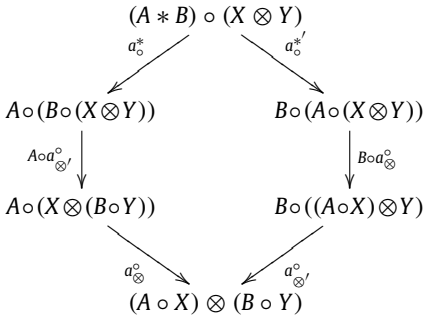
[Associativity.] The following diagrams are ones linking the various associativity morphisms. There are four pentagon-shaped diagrams which must commute.



These result in the following equations:

$$
\begin{array}{lll}
(11) & (a_* \circ X);\, a^*_\circ;\, (A \circ a^*_\circ) = a^*_\circ;\, a^*_\circ & (A \bullet a^*_\bullet);\, a^*_\bullet;\, (a_*^{-1} \bullet X) = a^*_\bullet;\, a^*_\bullet \\
 & (a_*^{-1} \circ X);\, a^{*'}_\circ;\, (A \circ a^{*'}_\circ) = a^{*'}_\circ;\, a^{*'}_\circ & (A \bullet a^{*'}_\bullet);\, a^{*'}_\bullet;\, (a_* \bullet X) = a^{*'}_\bullet;\, a^{*'}_\bullet \\
(12) & a^*_\circ;\, (A \circ d^\circ_\bullet);\, d^\circ_\bullet = d^\circ_\bullet;\, (C \bullet a^*_\circ) & d^\circ_\bullet;\, (A \bullet d^\circ_\bullet);\, a^*_\circ = C \circ a^*_\bullet;\, d^\bullet_\circ \\
 & a^{*'}_\circ;\, (A \circ d^\circ_\bullet);\, d^\circ_\bullet = d^\circ_\bullet;\, (C \bullet a^{*'}_\circ) & d^\circ_\bullet;\, (A \bullet d^\circ_\bullet);\, a^{*'}_\circ = C \circ a^*_\bullet;\, d^\bullet_\circ \\
(13) & a^*_\circ;\, (A \circ a^\circ_\otimes);\, (a^\circ_\otimes) = (a^\circ_\otimes);\, (a^*_\circ \otimes Y) & a^\bullet_\oplus;\, (A \bullet a^\bullet_\oplus);\, a^*_\bullet = (a^*_\bullet \oplus Y);\, a^\bullet_\oplus \\
 & a^{*'}_\circ;\, (A \circ a^\circ_\otimes);\, (a^\circ_\otimes) = (a^\circ_\otimes);\, (a^{*'}_\circ \otimes Y) & a^\bullet_\oplus;\, (A \bullet a^\bullet_\oplus);\, a^{*'}_\bullet = (a^{*'}_\bullet \oplus Y);\, a^\bullet_\oplus \\
 & a^*_\circ;\, (A \circ a^\circ_{\otimes'});\, (a^\circ_{\otimes'}) = (a^\circ_{\otimes'});\, (Y \otimes a^*_\circ) & a^\bullet_{\oplus'};\, (A \bullet a^\bullet_{\oplus'});\, a^*_\bullet = (Y \oplus a^*_\bullet);\, a^\bullet_{\oplus'} \\
 & a^{*'}_\circ;\, (A \circ a^\circ_{\otimes'});\, (a^\circ_{\otimes'}) = (a^\circ_{\otimes'});\, (Y \otimes a^{*'}_\circ) & a^\bullet_{\oplus'};\, (A \bullet a^\bullet_{\oplus'});\, a^{*'}_\bullet = (Y \oplus a^{*'}_\bullet);\, a^\bullet_{\oplus'} \\
(14) & a^\circ_\otimes;\, (a^\circ_\otimes \otimes Z);\, a_\otimes = (A \circ a_\otimes);\, a^\circ_\otimes & a_\oplus;\, (a^\bullet_\oplus \otimes Z);\, a^\bullet_\oplus = a^\bullet_\oplus;\, (A \bullet a_\oplus) \\
 & a^\circ_{\otimes'};\, (Z \otimes a^\circ_{\otimes'});\, a^{-1}_\otimes = (A \circ a^{-1}_\otimes);\, a^\circ_{\otimes'} & a^{-1}_\oplus;\, (Z \otimes a^\bullet_{\oplus'});\, a^\bullet_{\oplus'} = a^\bullet_{\oplus'};\, (A \bullet a^{-1}_\oplus)
\end{array}
$$

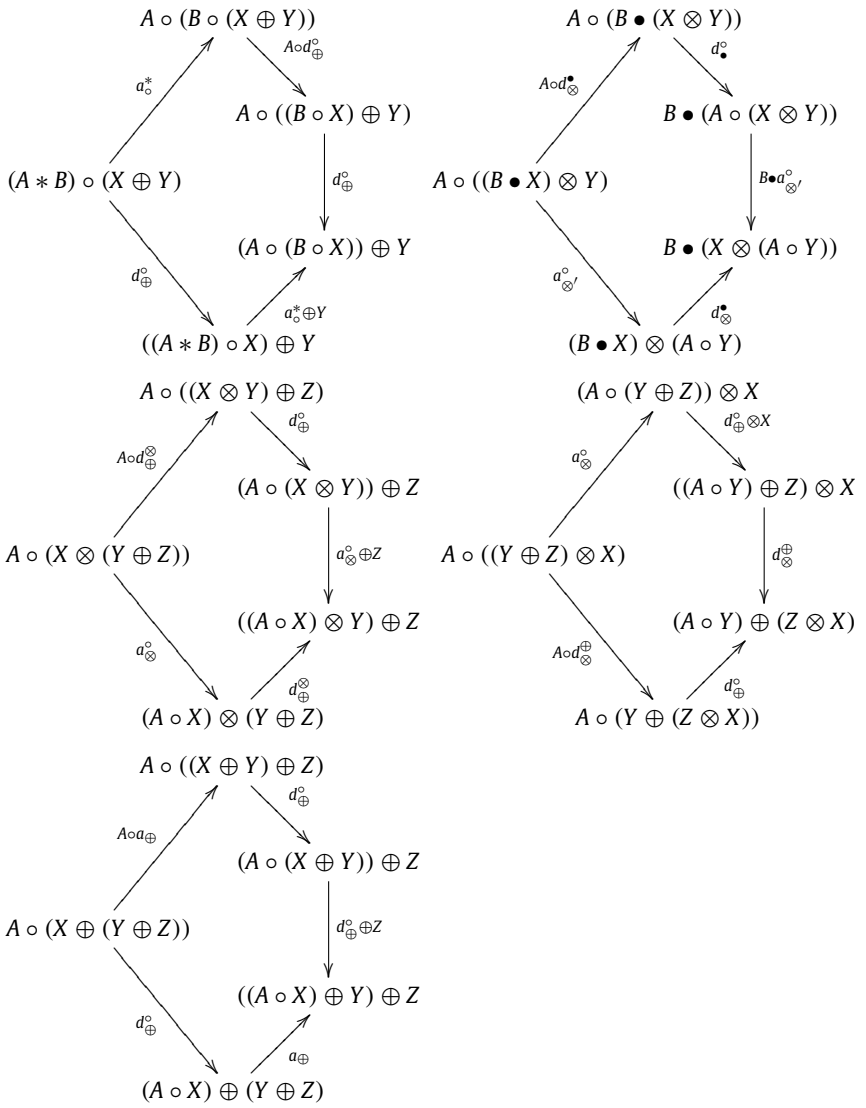
There is additionally a hexagon-shaped diagram which must commute



which results in the equations:

$$
\begin{array}{ll}
a^*_\circ;\, (A \circ a^\circ_{\otimes'});\, a^\circ_\otimes = a^{*'}_\circ;\, (B \circ a^\circ_\otimes);\, a^\circ_{\otimes'} & a^\bullet_\oplus;\, (A \bullet a^\bullet_{\oplus'});\, a^*_\bullet = a^{*'}_\bullet;\, (B \bullet a^\bullet_\oplus);\, a^\bullet_{\oplus'} \\
a^*_\circ;\, (A \circ a^\circ_\otimes);\, a^\circ_{\otimes'} = a^{*'}_\circ;\, (B \circ a^\circ_{\otimes'});\, a^\circ_\otimes & a^\bullet_{\oplus'};\, (A \bullet a^\bullet_\oplus);\, a^*_\bullet = a^{*'}_\bullet;\, (B \bullet a^\bullet_{\oplus'});\, a^\bullet_\oplus
\end{array}
\qquad (15)
$$

[Distributivity and associativity.] Diagrams made up of the distributivity morphisms. There are five pentagon-shaped diagrams which must commute.



These result in the following equations:

(16) $a^*_\circ;(A\circ d^\circ_\oplus);d^\circ_\oplus = d^\circ_\oplus;(a^*_\circ \oplus Y)$ $\quad d^\bullet_\otimes;(A\bullet d^\bullet_\otimes);a^*_\bullet = (a^*_\bullet \otimes Y);d^\bullet_\otimes$

$a^{*'}_\circ;(A\circ d^\circ_\oplus);d^\circ_\oplus = d^\circ_\oplus;(a^{*'}_\circ \oplus Y)$ $\quad d^\bullet_\otimes;(A\bullet d^\bullet_\otimes);a^{*'}_\bullet = (a^{*'}_\bullet \otimes Y);d^\bullet_\otimes$

$a^*_\circ;(A\circ d^\circ_{\oplus'});d^\circ_{\oplus'} = d^\circ_{\oplus'};(Y\oplus a^*_\circ)$ $\quad d^\bullet_{\otimes'};(A\bullet d^\bullet_{\otimes'});a^*_\bullet = (Y\otimes a^*_\bullet);d^\bullet_{\otimes'}$

$a^{*'}_\circ;(A\circ d^\circ_{\oplus'});d^\circ_{\oplus'} = d^\circ_{\oplus'};(Y\oplus a^{*'}_\circ)$ $\quad d^\bullet_{\otimes'};(A\bullet d^\bullet_{\otimes'});a^{*'}_\bullet = (Y\otimes a^{*'}_\bullet);d^\bullet_{\otimes'}$

(17) $(A\circ d^\bullet_\otimes);d^\circ_\bullet;(B\bullet a^\circ_{\otimes'}) = a^\circ_{\otimes'};d^\bullet_\otimes$ $\quad (B\circ a^\bullet_{\oplus'});d^\circ_\bullet;(A\bullet d^\circ_\oplus) = d^\circ_\oplus;a^\bullet_{\oplus'}$

$(A\circ d^\bullet_{\otimes'});d^\circ_\bullet;(B\bullet a^\circ_\otimes) = a^\circ_\otimes;d^\bullet_{\otimes'}$ $\quad (B\circ a^\bullet_\oplus);d^\circ_\bullet;(A\bullet d^\circ_{\oplus'}) = d^\circ_{\oplus'};a^\bullet_\oplus$

(18) $(A\circ d^\otimes_\oplus);d^\circ_\oplus;(a^\circ_\otimes\oplus Z) = a^\circ_\otimes;d^\otimes_\oplus$ $\quad (a^\bullet_\oplus\otimes Z);d^\bullet_\otimes;(A\bullet d^\oplus_\otimes) = d^\oplus_\otimes;a^\bullet_\oplus$

$(A\circ d^{\oplus'}_\otimes);d^\circ_\oplus;(a^\circ_{\otimes'}\oplus Z) = a^\circ_{\otimes'};d^{\oplus'}_\otimes$ $\quad (a^\bullet_{\oplus'}\otimes Z);d^\bullet_\otimes;(A\bullet d^{\oplus'}_\otimes) = d^{\oplus'}_\otimes;a^\bullet_{\oplus'}$

$(A\circ d^{\otimes'}_\oplus);d^\circ_{\oplus'};(Z\oplus a^\circ_\otimes) = a^\circ_\otimes;d^{\otimes'}_\oplus$ $\quad (Z\otimes a^\bullet_\oplus);d^\bullet_{\otimes'};(A\bullet d^{\otimes'}_\oplus) = d^{\otimes'}_\oplus;a^\bullet_\oplus$

$(A\circ d^\oplus_\otimes);d^\circ_{\oplus'};(Z\oplus a^\circ_{\otimes'}) = a^\circ_{\otimes'};d^\oplus_\otimes$ $\quad (Z\otimes a^\bullet_{\oplus'});d^\bullet_{\otimes'};(A\bullet d^\otimes_\oplus) = d^\otimes_\oplus;a^\bullet_{\oplus'}$

(19) $a^\circ_\otimes;(d^\circ_\oplus\otimes X);d^\oplus_\otimes = (A\circ d^\oplus_\otimes);d^\circ_\oplus$ $\quad d^\otimes_\oplus;(d^\bullet_\otimes\oplus X);a^\bullet_\oplus = d^\bullet_\otimes;(A\bullet d^\otimes_\oplus)$

$a^\circ_{\otimes'};(X\otimes d^\circ_\oplus);d^{\otimes'}_\oplus = (A\circ d^{\otimes'}_\oplus);d^\circ_\oplus$ $\quad d^{\otimes'}_\oplus;(X\oplus d^\bullet_\otimes);a^\bullet_{\oplus'} = d^\bullet_\otimes;(A\bullet d^{\otimes'}_\oplus)$

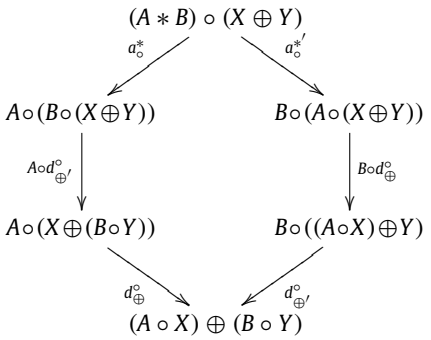
$a^\circ_\otimes;(d^\circ_{\oplus'}\otimes X);d^{\oplus'}_\otimes = (A\circ d^{\oplus'}_\otimes);d^\circ_{\oplus'}$ $\quad d^{\oplus'}_\otimes;(d^\bullet_{\otimes'}\oplus X);a^\bullet_\oplus = d^\bullet_{\otimes'};(A\bullet d^{\oplus'}_\otimes)$

$a^\circ_{\otimes'};(X\otimes d^\circ_{\oplus'});d^\otimes_\oplus = (A\circ d^\otimes_\oplus);d^\circ_{\oplus'}$ $\quad d^\oplus_\otimes;(X\oplus d^\bullet_{\otimes'});a^\bullet_{\oplus'} = d^\bullet_{\otimes'};(A\bullet d^\oplus_\otimes)$

(20) $(A\circ a_\oplus);d^\circ_\oplus;(d^\circ_\oplus\oplus Z) = d^\circ_\oplus;a_\oplus$ $\quad (d^\bullet_\otimes\otimes Z);d^\bullet_\otimes;(A\bullet a_\otimes) = a_\otimes;d^\bullet_\otimes$

$(A\circ a^{-1}_\oplus);d^\circ_{\oplus'};(Z\oplus d^\circ_{\oplus'}) = d^\circ_{\oplus'};a^{-1}_\oplus$ $\quad (Z\otimes d^\bullet_{\otimes'});d^\bullet_{\otimes'};(A\bullet a^{-1}_\otimes) = a^{-1}_\otimes;d^\bullet_{\otimes'}$

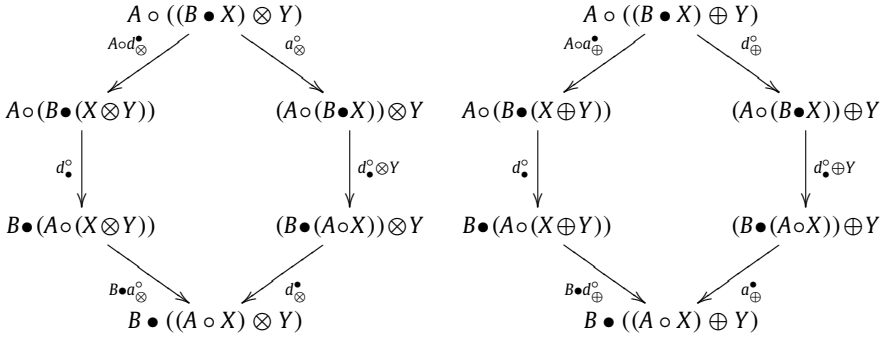
There are additionally three hexagon-shaped diagram which must commute. The first is



which results in the equations:

$$a^*_\circ;\,(A\circ d^\circ_{\oplus'});\,d^\circ_\oplus = a^{*'}_\circ;\,(B\circ d^\circ_\oplus);\,d^\circ_{\oplus'} \qquad d^\bullet_\otimes;\,(A\bullet d^\bullet_{\otimes'});\,a^*_\bullet = d^\bullet_{\otimes'};\,(B\bullet d^\bullet_\otimes);\,a^{*'}_\bullet \tag{21}$$
$$a^*_\circ;\,(A\circ d^\circ_\oplus);\,d^\circ_{\oplus'} = a^{*'}_\circ;\,(B\circ d^\circ_{\oplus'});\,d^\circ_\oplus \qquad d^\bullet_{\otimes'};\,(A\bullet d^\bullet_\otimes);\,a^*_\bullet = d^\bullet_\otimes;\,(B\bullet d^\bullet_{\otimes'});\,a^{*'}_\bullet$$

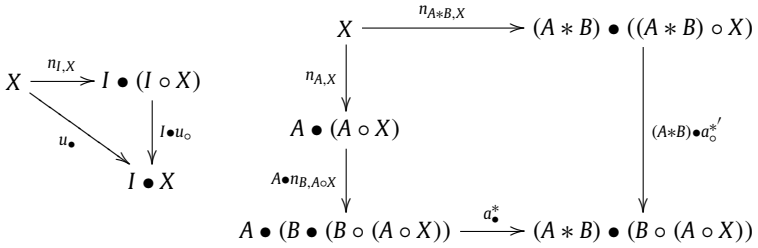
The remaining two are

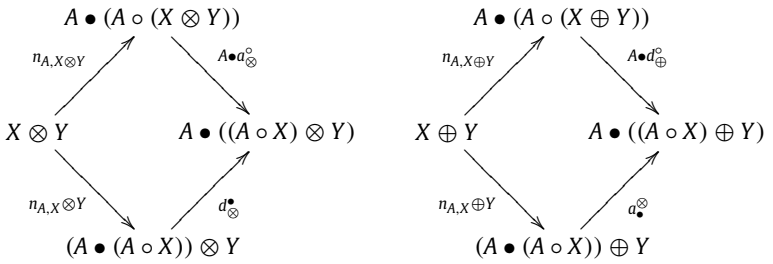


and these result in the equations:

$$a^\circ_\otimes;\,(d^\circ_\bullet\otimes Y);\,d^\bullet_\otimes = (A\circ d^\bullet_\otimes);\,d^\circ_\bullet;\,(B\bullet a^\circ_\otimes) \tag{22}$$
$$d^\circ_\oplus;\,(d^\circ_\bullet\oplus Y);\,a^\bullet_\oplus = (B\circ a^\bullet_\oplus);\,d^\circ_\bullet;\,(A\bullet d^\circ_\oplus)$$
$$a^\circ_{\otimes'};\,(Y\otimes d^\circ_\bullet);\,d^\bullet_{\otimes'} = (A\circ d^\bullet_{\otimes'});\,d^\circ_\bullet;\,(B\bullet a^\circ_{\otimes'})$$
$$d^\circ_{\oplus'};\,(Y\oplus d^\circ_\bullet);\,a^\bullet_{\oplus'} = (B\circ a^\bullet_{\oplus'});\,d^\circ_\bullet;\,(A\bullet d^\circ_{\oplus'})$$
$$d^\circ_\oplus;\,(d^\circ_\bullet\oplus Y);\,a^\bullet_\oplus = (A\circ a^\bullet_\oplus);\,d^\circ_\bullet;\,(B\bullet d^\circ_\oplus) \tag{23}$$
$$a^\circ_\otimes;\,(d^\circ_\bullet\otimes Y);\,d^\bullet_\otimes = (B\circ d^\bullet_\otimes);\,d^\circ_\bullet;\,(A\bullet a^\circ_\otimes)$$
$$d^\circ_{\oplus'};\,(Y\oplus d^\circ_\bullet);\,a^\bullet_{\oplus'} = (A\circ a^\bullet_{\oplus'});\,d^\circ_\bullet;\,(B\bullet d^\circ_{\oplus'})$$
$$a^\circ_{\otimes'};\,(Y\otimes d^\circ_\bullet);\,d^\bullet_{\otimes'} = (B\circ d^\bullet_{\otimes'});\,d^\circ_\bullet;\,(A\bullet a^\circ_{\otimes'})$$

[*n* and *e*.] Finally, there are diagrams linking the unit and counit of the $A\circ - \dashv A\bullet -$ adjunction with I and the associativity and distributivity morphisms.





With the symmetries this gives the equations:

$$
\begin{array}{lcc}
(24) & n_{I,X};\, I \bullet u_\circ = u_\bullet & e_{I,X};\, u_\bullet = u_\circ \\
(25) & n_{A*B,X};\, (A * B) \bullet a^{*'}_\circ = n_{A,X};\, A \bullet n_{B,A\circ X};\, a^*_\bullet & a^{*'}_\bullet;\, e_{A*B,X} = a^*_\circ;\, A \circ e_{B,A\bullet X};\, e_{A,X} \\
(26) & n_{X\otimes Y};\, (A \bullet a^\circ_\otimes) = (n_X \otimes Y);\, d^\bullet_\otimes & (A \circ a^\bullet_\oplus);\, e_{X\oplus Y} = d^\circ_\oplus;\, (e_X \oplus Y) \\
 & n_{Y\otimes X};\, (A \bullet a^\circ_{\otimes'}) = (Y \otimes n_X);\, d^\bullet_{\otimes'} & (A \circ a^\bullet_{\oplus'});\, e_{Y\oplus X} = d^\circ_{\oplus'};\, (Y \oplus e_X) \\
(27) & n_{X\oplus Y};\, (A \bullet d^\circ_\oplus) = (n_X \oplus Y);\, a^\bullet_\oplus & (A \circ d^\bullet_\otimes);\, e_{X\otimes Y} = a^\circ_\otimes;\, (e_X \otimes Y) \\
 & n_{Y\oplus X};\, (A \bullet d^\circ_{\oplus'}) = (Y \oplus n_X);\, a^\bullet_{\oplus'} & (A \circ d^\bullet_{\otimes'});\, e_{Y\otimes X} = a^\circ_{\otimes'};\, (Y \otimes e_X)
\end{array}
$$

In the next section we will explore these diagrams in more detail.

Example 4.2. (1) Given any monoidal closed category, regarding it as a compact linear distributive category it acts on itself via

$$A \circ B = A \otimes B \qquad \text{and} \qquad A \bullet B = A \multimap B.$$

This fails in general to give a linear actegory as the isomorphism $a^\bullet_\oplus : (A \bullet X) \oplus Y \longrightarrow A \bullet (X \oplus Y)$ is absent. However if one restricts the action to the compact objects (i.e., those objects for which the natural map $(A \multimap I) \otimes B \longrightarrow A \multimap B$ is an isomorphism) then this becomes a linear actegory with actions as above.

(2) Compact closed categories are, of course, a source of examples of the above. They are (compact) $*$-autonomous categories and so a bridge to the next example. They are important as not only are they the foundation for what Girard calls the "geometry of interaction", but also for a family of compact closed categories (essentially span categories) which were studied by Abramsky [2,3] under the name of "interaction categories". These used explicit ideas from process calculus to give a categorical semantics for processes. Acting on themselves these give examples of linear actegories. Abramsky, Gay, and Nagarajan also considered adding "specifications" to these categories [4] which made them (mix) $*$-autonomous categories and, thus, less degenerate models of linear actegories.

(3) Given any linear distributive category $\mathscr{X}$, the objects which have linear adjoints (complements) form a $*$-autonomous subcategory $\mathrm{Map}(\mathscr{X})$. There is an obvious action of $\mathrm{Map}(\mathscr{X})$ on $\mathscr{X}$ defined by $A \circ X = A \otimes X$ and $A \bullet X = A^* \oplus X$ (much as in the first example). In particular, a $*$-autonomous category acting in the obvious manner on itself is a linear actegory.

(4) None of the above examples illustrate well the separation of messages from the message-passing. However, using Benton's approach [9] to models of linear logic (with exponentials), which links the intuitionistic terms to the linear terms by a monoidal adjunction, gives an important model of a linear actegory where such separation is displayed.

This model was used by Barber et al. [6] to provide a semantics for Milner's action calculus. As the action calculus was developed, in part, to provide a semantic framework for systems such as the π-calculus this suggests that there is a very close connection between the work of Barber et al. [6] and what is being proposed here. This is indeed the case, however, there are also some important differences. A model of their logic is an example of a linear actegory only when their monoidal closed category is actually a $*$-autonomous category. In this regard linear actegories demand *more* structure. On the other hand, not every linear actegory arises in this manner. One very obvious reason is that in their setting the messages and the processes are still very closely linked by the adjoint and it is thus possible to turn process code into message code. While often in practice this may be a desirable feature, it may also be something that one does not want to allow. In our models we do not assume such a connection exists.

If $\mathscr{X}$ is a $*$-autonomous category with an exponential comonad $! : \mathscr{X} \longrightarrow \mathscr{X}$ then, as described by Benton [9], the comonad induces a monoidal adjunction $V \vdash W : \mathscr{X} \longrightarrow \mathscr{X}_!$, where essentially $W = !$ and $\mathscr{X}_!$ is a cartesian closed category. The action $\circ : \mathscr{X}_! \times \mathscr{X} \longrightarrow \mathscr{X}$ is

$$X \circ Y = W(X) \otimes Y$$

and as $W(X \times Y) \cong W(X) \otimes W(Y)$ this automatically gives an action. The action $\bullet : (\mathscr{X}_!)^{\mathrm{op}} \times \mathscr{X} \longrightarrow \mathscr{X}$ is given by

$$X \bullet Y = W(X)^* \oplus Y.$$

That this is a linear actegory is now a straightforward, if lengthy, exercise.

5. Categorical semantics

We shall say that a linear $\mathscr{A}$-actegory is $\mathscr{A}$-additive in case the monoidal category $\mathscr{A}$ is a distributive monoidal category (i.e., it has coproducts over which the tensor distributes) and the covariant action preserves these coproducts while the contravariant action turns them into products.

Our aim is to show that the proof theory of the message-passing logic, as represented by the terms, forms a linear additive actegory in the above sense. To achieve this we shall show in this section how one may collect the proof theory for message-passing into a linear additive actegory (completeness). In the next section we show that given any interpretation of the axioms into such a linear actegory one can extend the interpretation to the whole message logic (soundness).

To preserve the sanity of reader and writer alike we shall present a recipe for these processes exemplifying only some of the details. These matters have already been well-explored for the message fragment and the linear distributive fragment. Accordingly, when it comes to the details we shall focus on the actions. We begin by proving completeness. This involves showing:

(a) The proofs of the message-passing logic sequents with empty context and one input and output type

$$\emptyset \mid X \Vdash Y$$

form a linear distributive category.

(b) The proofs of the message logic with one input (and necessarily one output)

$$x : A \vdash s : B$$

form a distributive monoidal category.

(c) The two required actions of a linear actegory are present together with the coherence maps and that they satisfy all the required coherences of the previous section.

Parts (a) and (b) have been established in [12] and [16] respectively, so we shall concentrate our efforts on (c). To begin this proof we need to establish the functorial nature of the actions. This, in turn, will lead into the naturality of the coherence maps and establishing the coherences.

5.1. The actions are functors

Here is the definition of the action $\circ : \mathscr{A} \times \mathscr{X} \longrightarrow \mathscr{X}$ built from an arbitrary monoidal map $f : A \longrightarrow B$ and a process $s : X \longrightarrow Y$.

$$\dfrac{\dfrac{x : A \vdash f : B \qquad s :: \alpha : X \Vdash \beta : Y}{\beta[f] \cdot s \;::\; x : A \mid \alpha : X \Vdash \beta : B \circ Y}}{\alpha\langle x\rangle \cdot \beta[f] \cdot s \;::\; \emptyset \mid \alpha : A \circ X \Vdash \beta : B \circ Y}$$

In order to show that this is a functor we must show that it preserves composition. As composition amounts to a cut, this amounts to showing that cuts inside the functor can be equivalently expressed as a cut outside. Here is the calculation in reverse.

$$\begin{aligned}
\alpha\langle x\rangle \cdot \beta[f] \cdot s \,{}_\beta;_\gamma\, \gamma\langle y\rangle \cdot \delta[g] \cdot t &\Longrightarrow \alpha\langle x\rangle \cdot (\beta[f] \cdot s \,{}_\beta;_\gamma\, \gamma\langle y\rangle \cdot \delta[g] \cdot t) \\
&\Longrightarrow \alpha\langle x\rangle \cdot (y \mapsto s \,{}_\beta;_\gamma\, \delta[g] \cdot t)f \\
&\Longrightarrow \alpha\langle x\rangle \cdot (y \mapsto \delta[g] \cdot (s \,{}_\beta;_\gamma\, t))f \\
&\Longrightarrow \alpha\langle x\rangle \cdot \delta[(y \mapsto g)f] \cdot (s \,{}_\beta;_\gamma\, t)
\end{aligned}$$

Thus $\circ$ preserves composition. As we shall see shortly, in the next section, it preserves identities.

The symmetry which is embodied in the term logic means that the action $\bullet : \mathscr{A}^{\mathrm{op}} \times \mathscr{X} \longrightarrow \mathscr{X}$ has an identical term though the arrangement of the types is different.

$$\dfrac{\dfrac{x : A \vdash f : B \qquad s :: \alpha : X \Vdash \beta : Y}{\alpha[f] \cdot s \;::\; x : A \mid \alpha : B \bullet X \Vdash \beta : Y}}{\beta\langle x\rangle \cdot \alpha[f] \cdot s \;::\; \emptyset \mid \alpha : B \bullet X \Vdash \beta : A \bullet Y}$$

This means the preservation of composition (and identities) for $\bullet$ is essentially the same proof.

5.2. Identities

We shall denote the identity map on a type by

$$\alpha =_X \beta :: \alpha : X \longrightarrow \beta : X.$$

However, its definition as a term depends on the type X. If the type is primitive then this identity is built-in and defined to behave in the correct manner. However, if X is not primitive we must provide an inductive definition.

$$\begin{aligned}
\alpha =_X \beta \quad &\text{is} \quad \alpha =_X \beta \qquad \text{for } X \text{ a primitive type}\\
\alpha =_{X\otimes Y} \beta \quad &\text{is} \quad \alpha\langle\alpha_1,\alpha_2\rangle \cdot \beta\begin{bmatrix}\beta_1 \mapsto \alpha_1 =_X \beta_1\\ \beta_2 \mapsto \alpha_2 =_Y \beta_2\end{bmatrix}\\
\alpha =_{X\oplus Y} \beta \quad &\text{is} \quad \beta\langle\beta_1,\beta_2\rangle \cdot \alpha\begin{bmatrix}\alpha_1 \mapsto \alpha_1 =_X \beta_1\\ \alpha_2 \mapsto \alpha_2 =_Y \beta_2\end{bmatrix}\\
\alpha =_{\top} \beta \quad &\text{is} \quad \alpha\langle\,\rangle \cdot \beta[\,]\\
\alpha =_{\bot} \beta \quad &\text{is} \quad \beta\langle\,\rangle \cdot \alpha[\,]\\
\alpha =_{A\circ X} \beta \quad &\text{is} \quad \alpha\langle x\rangle \cdot \beta[\iota(x)] \cdot \alpha =_X \beta\\
\alpha =_{A\bullet X} \beta \quad &\text{is} \quad \beta\langle x\rangle \cdot \alpha[\iota(x)] \cdot \alpha =_X \beta
\end{aligned}$$

Here $x : A \vdash \iota(x) : A$ is the identity map in the message logic for the type A (where x here stands for a pattern in general). Notice also that this definition confirms that the functors (above) preserve identities. Clearly we also need an inductive definition of the identities in the message logic:

$$\begin{aligned}
x : A \vdash \iota(x) : A \ \text{ is } \ x : A \vdash x : A \qquad &\text{for } A \text{ a primitive type}\\
(x,y) : A * B \vdash \iota(x,y) : A * B \ \text{ is } \ (x,y) : A * B \vdash (\iota(x),\iota(y)) : A * B &\\
z : A + B \vdash \iota(z) : A + B \ \text{ is } \ z : A + B \vdash \begin{Bmatrix}\sigma_1(x) \mapsto \sigma_0(\iota(x))\\ \sigma_2(y) \mapsto \sigma_1(\iota(y))\end{Bmatrix} z : A + B &
\end{aligned}$$

It remains to do an inductive proof that these terms do act as identity maps. We shall focus on the step for the covariant action to give a feel of how this inductive proof (which is straightforward) plays out. Consider

$$\alpha\langle x\rangle \cdot \beta[\iota(x)] \cdot \alpha =_X \beta \ ;_\beta \ s.$$

We do an induction on the structure of the term s. There are two basic cases: either the leading structure of s interacts along the channel β or it does not. If it does not then this allows us to push the identity term inside the leading structure and to invoke the inductive hypothesis. This leaves the case in which there is interaction on β and this means s must be of the form $\beta\langle y\rangle \cdot s'$ and we have

$$\begin{aligned}
\alpha\langle x\rangle \cdot \beta[\iota(x)] \cdot \alpha =_X \beta \ ;_\beta \beta\langle y\rangle \cdot s' \ &\Longrightarrow \ \alpha\langle x\rangle \cdot (y \mapsto \alpha =_X \beta \ ;_\beta s')\iota(x)\\
&\Longrightarrow \ \alpha\langle y\rangle \cdot \alpha =_X \beta \ ;_\beta s'\\
&\Longrightarrow \ \alpha\langle y\rangle \cdot s'
\end{aligned}$$

where we invoke the induction hypothesis to obtain the last step.

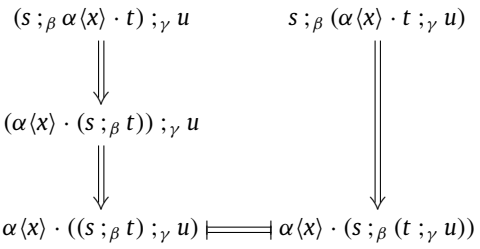
5.3. Associativity and interchange

The associativity of composition, as represented by a cut, must also be proven. The proof is again an inductive argument concerning the cut elimination process. Here we highlight some of the inductive steps of this argument which involve the action $\circ : \mathscr{A} \times \mathscr{X} \longrightarrow \mathscr{X}$. We consider the cases determined by terms which have their leading action on this type. This means the term is either of the form $\alpha\langle x\rangle \cdot s$ or $\alpha[f] \cdot s$. We consider the first case in more detail. In a sequence of three cuts such a term can occur of course in three positions. For whichever position it occurs in it is shown that the inductive hypothesis may be used on a combination of smaller terms to show that the original term is associative.

Consider the case when it is in the first position and the cut does not occur on α. The following diagram of cut-elimination rewrites, where the bottom equality uses the inductive hypothesis proves associativity in this case.

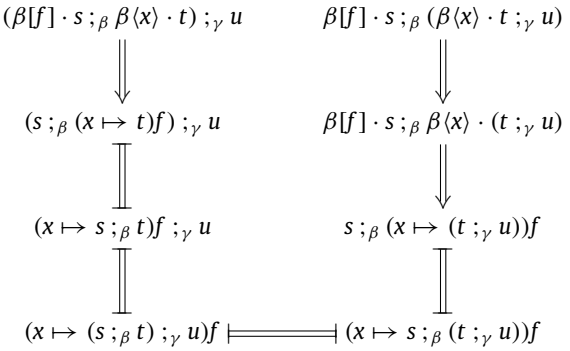
$$\begin{array}{ccc}
(\alpha\langle x\rangle \cdot s \ ;_\beta t) \ ;_\gamma u & & \alpha\langle x\rangle \cdot s \ ;_\beta (t \ ;_\gamma u)\\
\Downarrow & & \Downarrow\\
(\alpha\langle x\rangle \cdot (s \ ;_\beta t)) \ ;_\gamma u & & \Downarrow\\
\Downarrow & & \Downarrow\\
\alpha\langle x\rangle \cdot ((s \ ;_\beta t) \ ;_\gamma u) & \Longleftrightarrow & \alpha\langle x\rangle \cdot (s \ ;_\beta (t \ ;_\gamma u))
\end{array}$$

A very similar argument holds for the middle position provided α is not the interacting channel determined by the leftmost cut.



Now if α is the channel of the leftmost cut then either the leftmost terms leading action is on that channel or not. If it is not, we can move the cut inside the action. Now provided the second cut (on γ) is also not on that channel this can be moved inside the action and we can then invoke the inductive hypothesis. Fortunately, due to the way a cut works, it is impossible for the outer terms to share a channel, thus the second cut is guaranteed to be independent of this action.

This leaves the case when the leftmost terms leading action is on the channel. The following diagram then proves this case.



Similar arguments are now easily inferred for the term in the last position.

5.4. The natural transformations

At this stage we have demonstrated that we have the basic functorial data for a linear actegory, namely a (distributive) monoidal category acting on a linearly distributive category in a covariant and contravariant way. It remains to show that the coherent transformations are present and satisfy the required conditions.

To accomplish this task we shall indicate the definition of the natural transformations and illustrate how one establishes their naturality. Here is how one natural transformation from each of the major symmetry classes is defined in the term logic:

$$r_{\oplus} := \beta\langle\beta_1, \beta_2\rangle \cdot \beta_2\langle\,\rangle \cdot \alpha =_X \beta_1$$

$$a_{\otimes} := \alpha\langle\alpha_1, \alpha_2\rangle \cdot \alpha_1\langle\alpha_{11}, \alpha_{12}\rangle \cdot \beta\begin{bmatrix}\beta_1 \mapsto \alpha_{11} =_X \beta_1\\ \beta_2 \mapsto \beta_2\begin{bmatrix}\beta_{21} \mapsto \alpha_{12} =_Y \beta_{21}\\ \beta_{22} \mapsto \alpha_2 =_Z \beta_{22}\end{bmatrix}\end{bmatrix}$$

$$l_{\otimes} := \alpha\langle\alpha_1, \alpha_2\rangle \cdot \alpha_1\langle\,\rangle \cdot \alpha_2 =_X \beta$$

$$r_{\otimes} := \alpha\langle\alpha_1, \alpha_2\rangle \cdot \alpha_2\langle\,\rangle \cdot \alpha_1 =_X \beta$$

$$c_{\otimes} := \alpha\langle\alpha_1, \alpha_2\rangle \cdot \beta\begin{bmatrix}\beta_1 \mapsto \alpha_2 =_Y \beta_1\\ \beta_2 \mapsto \alpha_1 =_X \beta_2\end{bmatrix}$$

$$u_{\circ} := \alpha\langle()\rangle \cdot \alpha =_X \beta$$

$$a^*_{\circ} := \alpha\langle(x, y)\rangle \cdot \beta[x] \cdot \beta[y] \cdot \alpha =_X \beta$$

$$a^{\circ}_{\otimes} := \alpha\langle x\rangle \cdot \alpha\langle\alpha_1, \alpha_2\rangle \cdot \beta\begin{bmatrix}\beta_1 \mapsto \beta_1[x] \cdot \alpha_1 =_X \beta_1\\ \beta_2 \mapsto \alpha_2 =_X \beta_2\end{bmatrix}$$

$$d^{\otimes}_{\oplus} := \beta\langle\beta_1, \beta_2\rangle \cdot \alpha\langle\alpha_1, \alpha_2\rangle \cdot \alpha_2\begin{bmatrix}\alpha_{21} \mapsto \beta\begin{bmatrix}\beta_1 \mapsto \alpha_1 =_X \beta_{11}\\ \beta_2 \mapsto \alpha_{21} =_Y \beta_{12}\end{bmatrix}\\ \alpha_{22} \mapsto \alpha_{22} =_Z \beta_2\end{bmatrix}$$

$$d^\circ_\oplus := \beta\langle\beta_1,\beta_2\rangle \cdot \alpha\langle x\rangle \cdot \beta_1[x] \cdot \alpha\begin{bmatrix}\alpha_1 \mapsto \alpha_1 =_X \beta_1\\ \alpha_2 \mapsto \alpha_2 =_X \beta_2\end{bmatrix}$$
$$d^\circ_\bullet := \alpha\langle x\rangle \cdot \beta\langle y\rangle \cdot \beta[x] \cdot \alpha[y] \cdot \alpha_1 =_X \beta_1$$
$$n := \beta\langle x\rangle \cdot \beta[x] \cdot \alpha =_X \beta$$

The natural transformations labelled with a should be isomorphisms. In particular $a^\circ_\otimes$ should be an isomorphism. It is not difficult to check that its inverse is given by

$$(a^\circ_\otimes)^{-1} := \alpha\langle\alpha_1,\alpha_2\rangle \cdot \alpha_1\langle x\rangle \cdot \beta[x] \cdot \beta\begin{bmatrix}\beta_1 \mapsto \alpha_1 =_X \beta_1\\ \beta_2 \mapsto \alpha_2 =_X \beta_2\end{bmatrix}.$$

It remains to check the naturality of these morphisms. We shall demonstrate what is involved for $a^\circ_\otimes$, i.e., given $f : A \longrightarrow B$, $s : W \longrightarrow Y$, and $t : X \longrightarrow Z$, that the following categorical diagram commutes.

$$\begin{array}{ccc} A \circ (W \otimes X) & \xrightarrow{a^\circ_\otimes} & (A \circ W) \otimes X \\ \Big\downarrow{\scriptstyle f\circ(s\otimes t)} & & \Big\downarrow{\scriptstyle (f\circ s)\otimes t} \\ B \circ (Y \otimes Z) & \xrightarrow{a^\circ_\otimes} & (B \circ Y) \otimes Z \end{array}$$

We shall translate the upper route into a term and show that (without looking into f, s, or t) we can manipulate it into a form which is equivalent to the lower route. When we translate the terms s and t we shall indicate that it runs from channel α to β (i.e., $s :: \alpha : W \longrightarrow \beta : Y$) by labelling it $s[\alpha;\beta]$. The top route then gives

$$\begin{aligned}
&\alpha\langle x\rangle \cdot \alpha\langle\alpha_1,\alpha_2\rangle \cdot \quad \beta\begin{bmatrix}\beta_1 \mapsto \beta_1[x] \cdot \alpha_1 =_W \beta_1\\ \beta_2 \mapsto \alpha_2 =_X \beta_2\end{bmatrix} ;_\beta \\
&\qquad\qquad \beta\langle\beta_1,\beta_2\rangle \cdot \gamma\begin{bmatrix}\gamma_1 \mapsto \beta_1\langle x\rangle \cdot \gamma_1[f(x)] \cdot s[\beta_1;\gamma_1]\\ \gamma_2 \mapsto t[\beta_2;\gamma_2]\end{bmatrix} \\
\Longrightarrow\ &\alpha\langle x\rangle \cdot \alpha\langle\alpha_1,\alpha_2\rangle \cdot (\beta_1[x] \cdot \alpha_1 =_W \beta_1 \ ;_{\beta_1} \alpha_2 =_X \beta_2 ;_{\beta_2} \\
&\qquad\qquad \gamma\begin{bmatrix}\gamma_1 \mapsto \beta_1\langle x\rangle \cdot \gamma_1[f(x)] \cdot s[\beta_1;\gamma_1]\\ \gamma_2 \mapsto t[\beta_2;\gamma_2]\end{bmatrix} \\
\Longrightarrow\ &\alpha\langle x\rangle \cdot \alpha\langle\alpha_1,\alpha_2\rangle \cdot \gamma\begin{bmatrix}\gamma_1 \mapsto \beta_1[x] \cdot \alpha_1 =_W \beta_1 ;_{\beta_1} \beta_1\langle x\rangle \cdot \gamma_1[f(x)] \cdot s[\beta_1;\gamma_1]\\ \gamma_2 \mapsto \alpha_2 =_X \beta_2 ;_{\beta_2} t[\beta_2;\gamma_2]\end{bmatrix} \\
\Longrightarrow\ &\alpha\langle x\rangle \cdot \alpha\langle\alpha_1,\alpha_2\rangle \cdot \gamma\begin{bmatrix}\gamma_1 \mapsto \gamma_1[f(x)] \cdot s[\alpha_1;\gamma_1]\\ \gamma_2 \mapsto t[\alpha_2;\gamma_2]\end{bmatrix},
\end{aligned}$$

and the bottom route

$$\begin{aligned}
&\alpha\langle x\rangle \cdot \beta[f(x)] \cdot \alpha\langle\alpha_1,\alpha_2\rangle \cdot \beta\begin{bmatrix}\beta_1 \mapsto s[\alpha_1;\beta_1]\\ \beta_2 \mapsto t[\alpha_2;\beta_2]\end{bmatrix} ;_\beta \\
&\qquad\qquad \beta\langle y\rangle \cdot \beta\langle\beta_1,\beta_2\rangle \cdot \gamma\begin{bmatrix}\gamma_1 \mapsto \gamma_1[y] \cdot \beta_1 =_Y \gamma_1\\ \gamma_2 \mapsto \beta_2 =_Z \gamma_2\end{bmatrix} \\
\Longrightarrow\ &\alpha\langle x\rangle \cdot \Bigg(y \mapsto \alpha\langle\alpha_1,\alpha_2\rangle \cdot \quad \beta\begin{bmatrix}\beta_1 \mapsto s[\alpha_1;\beta_1]\\ \beta_2 \mapsto t[\alpha_2;\beta_2]\end{bmatrix} ;_\beta \\
&\qquad\qquad \beta\langle\beta_1,\beta_2\rangle \cdot \gamma\begin{bmatrix}\gamma_1 \mapsto \gamma_1[y] \cdot \beta_1 =_Y \gamma_1\\ \gamma_2 \mapsto \beta_2 =_Z \gamma_2\end{bmatrix}\Bigg) f(x) \\
\Longrightarrow\ &\alpha\langle x\rangle \cdot \Bigg(\alpha\langle\alpha_1,\alpha_2\rangle \cdot \quad \beta\begin{bmatrix}\beta_1 \mapsto s[\alpha_1;\beta_1]\\ \beta_2 \mapsto t[\alpha_2;\beta_2]\end{bmatrix} ;_\beta \\
&\qquad\qquad \beta\langle\beta_1,\beta_2\rangle \cdot \gamma\begin{bmatrix}\gamma_1 \mapsto \gamma_1[f(x)] \cdot \beta_1 =_Y \gamma_1\\ \gamma_2 \mapsto \beta_2 =_Z \gamma_2\end{bmatrix}\Bigg) \\
\Longrightarrow\ &\alpha\langle x\rangle \cdot \alpha\langle\alpha_1,\alpha_2\rangle \cdot \Bigg(s[\alpha_1;\beta_1] ;_{\beta_1} \Bigg(t[\alpha_2;\beta_2] ;_{\beta_2} \gamma\begin{bmatrix}\gamma_1 \mapsto \gamma_1[f(x)] \cdot \beta_1 =_Y \gamma_1\\ \gamma_2 \mapsto \beta_2 =_Z \gamma_2\end{bmatrix}\Bigg)\Bigg) \\
\Longrightarrow\ &\alpha\langle x\rangle \cdot \alpha\langle\alpha_1,\alpha_2\rangle \cdot \gamma\begin{bmatrix}\gamma_1 \mapsto \gamma_1[f(x)] \cdot s[\alpha_1;\beta_1] ;_{\beta_1} \beta_1 =_Y \gamma_1\\ \gamma_2 \mapsto t[\alpha_2;\beta_2] ;_{\beta_2} \beta_2 =_Z \gamma_2\end{bmatrix} \\
\Longrightarrow\ &\alpha\langle x\rangle \cdot \alpha\langle\alpha_1,\alpha_2\rangle \cdot \gamma\begin{bmatrix}\gamma_1 \mapsto \gamma_1[f(x)] \cdot s[\alpha_1;\gamma_1]\\ \gamma_2 \mapsto t[\alpha_2;\gamma_2]\end{bmatrix}
\end{aligned}$$

which therefore proves the naturality of $a^\circ_\otimes$.

5.5. Completeness

In order to establish completeness of the logic it is now necessary to check that all the coherence diagrams commute. This is a lengthy exercise most of which we will leave to the reader!

We shall explicitly check the triangle equalities for the parameterised adjunction. Because of the symmetry it actually suffices to check just one:

$$A \circ n_X ; e_{A\circ X} = 1_{A\circ X} : A \circ X \longrightarrow A \circ X.$$

Here is the explicit calculation:

$$\begin{aligned}
&A \circ n_X \;;_\beta e_{A\circ X}\\
&\quad = \alpha\langle w\rangle \cdot \beta[w] \cdot \beta\langle x\rangle \cdot \beta[x] \cdot \alpha =_X \beta \;;_\beta\; \beta\langle y\rangle \cdot \beta[y] \cdot \beta\langle z\rangle \cdot \gamma[z] \cdot \beta =_X \gamma\\
&\Longrightarrow\; \alpha\langle w\rangle \cdot \big(\beta\langle x\rangle \cdot \beta[x] \cdot \alpha =_X \beta \;;_\beta\; \beta[w] \cdot \beta\langle z\rangle \cdot \gamma[z] \cdot \beta =_X \gamma\big)\\
&\Longrightarrow\; \alpha\langle w\rangle \cdot \big(\beta[w] \cdot \alpha =_X \beta \;;_\beta\; \beta\langle z\rangle \cdot \gamma[z] \cdot \beta =_X \gamma\big)\\
&\Longrightarrow\; \alpha\langle w\rangle \cdot \big(\alpha =_X \beta \;;_\beta\; \gamma[w] \cdot \beta =_X \gamma\big)\\
&\Longrightarrow\; \alpha\langle w\rangle \cdot \gamma[w] \cdot \alpha =_X \gamma = 1_{A\circ X}.
\end{aligned}$$

The other aspect of this setting we have not discussed is the coproducts. We expect that $(A + B) \circ X$ is the coproduct of $A \circ X$ and $B \circ X$. If we have proofs

$$s_1 :: \emptyset \mid \Gamma, \alpha : A \circ X \Vdash \Delta \qquad \text{and} \qquad s_2 :: \emptyset \mid \Gamma, \alpha : B \circ X \Vdash \Delta$$

it is not hard to see (especially using the representability results discussed below) that one can construct a proof

$$s :: \emptyset \mid \Gamma, \alpha : (A + B) \circ X \Vdash \Delta.$$

However if one has a proof s as above how does one see it as a cotuple of proofs?

Consider the effect of substituting on α the identity map of $(A + B) \circ X$ into s (where the first reduction is in reverse):

$$\begin{aligned}
s \;&\Longrightarrow\; \beta =_{(A+B)\circ X} \alpha \;;_\alpha s\\
&\Longrightarrow\; \beta\langle z\rangle \cdot \alpha \left[\begin{Bmatrix} \sigma_1(x) \mapsto \sigma_0(x) \\ \sigma_2(y) \mapsto \sigma_1(y) \end{Bmatrix} z \right] \cdot \beta =_X \alpha \;;_\alpha s\\
&\Longrightarrow\; \beta\langle z\rangle \cdot \begin{Bmatrix} \sigma_1(x) \mapsto \alpha[\sigma_0(x)] \cdot \beta =_X \alpha \;;_\alpha s \\ \sigma_2(y) \mapsto \alpha[\sigma_1(y)] \cdot \beta =_X \alpha \;;_\alpha s \end{Bmatrix}\\
&\Longrightarrow\; \beta\langle z\rangle \cdot \begin{Bmatrix} \sigma_1(x) \mapsto \alpha[\sigma_0(x)] \cdot s \\ \sigma_2(y) \mapsto \alpha[\sigma_1(y)] \cdot s \end{Bmatrix}
\end{aligned}$$

This shows how the term s can be decomposed as a cotuple of the composites with the injections. Thus $(A + B) \circ X$ is the coproduct of $A \circ X$ and $B \circ X$.

This completes our discussion of the completeness of the logic. We have shown that:

Proposition 5.1. *The terms of the message-passing logic between single types with cut as composition form a linear additive actegory.*

In fact, if $\mathscr{A}$ is any monoidal category this shows how we can construct a linear distributive category from $\mathscr{A}$ and the empty poly-$\mathscr{A}$-actegory (i.e., the initial linear $\mathscr{A}$-actegory). The result is, of course, definitely a non-empty linear actegory: the units $\top$ and $\bot$ must always be present which implies that $A \circ \bot$ and $A \bullet \top$ for all $A \in \mathscr{A}$ are non-trivial objects. It is also not hard to see that this will be a $*$-autonomous category as, inductively $(A \circ X)^* = A \bullet X^*$ and $(A \bullet Y)^* = A \circ Y^*$, with the base case the units.

6. Representability and soundness

In the theory of polycategories representability plays a crucial role in getting between the purely categorical (object-to-object) view and the circuit (polycategorical) view of the maps. In fact, to be the category of maps[2] of a representable polycategory is precisely to be a linear distributive category. Similarly to be the category of maps of a representable multicategory is precisely to be a monoidal category.

For the message-passing logic an exactly analogous correspondence holds:

Theorem 6.1. *To be the category of maps of a representable (additive) poly-actegory is precisely to be a linear (additive) actegory.*

[2] In this section and in the sequel we will call a polymap or multimap with singleton domain and singleton codomain simply a *map*.

Instead of a polycategory we must start with a polycategory with a multicategorical action: a *poly-actegory*. This makes the polymaps have a type which match the sequent structure of the message passing logic: composition in a poly-actegory is given by the two sorts of cut and the evident associativity and interchange laws must hold. Representability then entails the representability of all the features of the logic: the tensor, the par, and the two actions. To be a linear actegory is then precisely to be the maps in a representable poly-actegory.

The soundness of the term calculus is therefore determined by its soundness in any representable poly-actegory. However, the equations of the term calculus were developed from the proof theory in Section 2 and Section 3. Thus with the identification of the proof theory with poly-actegories we have the soundness by construction.

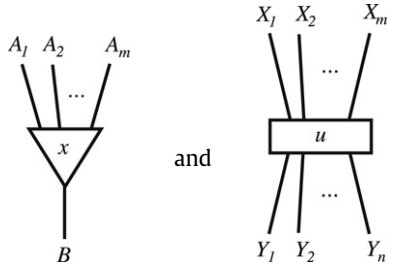
The purpose of this section is to introduce poly-actegories as a formulation of the proof theory of the message passing logic and to prove Theorem 6.1.

6.1. Poly-actegories and circuit representation

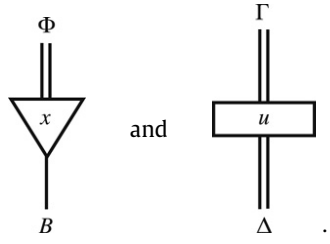
The proof theory for the two-sided cut rule lies in polycategories. Essentially they are the natural deduction style proofs for the system. Polycategories have an extremely intuitive representation using circuits (see [10]). This gives further indication that the proof theory and its categorical formulation is capturing a very natural phenomenon.

We wish to show how the proof theory of message-passing, as represented by poly-actegories, may also be represented using circuits. Every proof in multiplicative linear logic has a presentation as a circuit [10]. To extend this to the message-passing logic we must describe how the two action rules are to be interpreted using circuits.

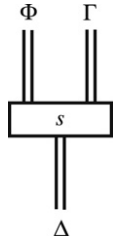
A multicategory $\mathscr{M}$ has objects $\mathrm{ob}(\mathscr{M})$ and multimaps $x : \Phi \longrightarrow B$ where $\Phi \subset \mathrm{ob}(\mathscr{M})$. A polycategory $\mathscr{P}$ has objects $\mathrm{ob}(\mathscr{P})$ and polymaps $u : \Gamma \longrightarrow \Delta$ where $\Gamma, \Delta \subset \mathrm{ob}(\mathscr{P})$. Multimaps and polymaps are represented in the circuit notation, where suppose $\Phi = A_1, \ldots, A_m$, $\Gamma = X_1, \ldots, X_m$, and $\Delta = Y_1, \ldots, Y_n$, as



respectively. Since we consider symmetric multicategories and symmetric polycategories this allows the "wires" to cross. Often, to simplify the circuit notation, we will use a double line to indicate a possibly empty set of wires as in



A *poly-actegory* $\mathscr{X}$ is a polycategory $\mathscr{P}$ acted on by a multicategory $\mathscr{M}$. What this means is that the polymaps in $\mathscr{X}$, instead of the usual polymaps as in $\mathscr{P}$ above, will contain inputs with types coming from both $\mathscr{P}$ and $\mathscr{M}$. A typical example is $s : \Phi \mid \Gamma \longrightarrow \Delta$. These sorts of polymaps may also be represented using the circuit notation as



where the type of a wire indicates whether it is from $\mathscr{M}$ or $\mathscr{P}$.

There are two types of composition in a poly-actegory. The first composes a multimap in $\mathscr{M}$ with a polymap in $\mathscr{X}$. Given a multimap $f : \Phi \longrightarrow A$ in $\mathscr{M}$ and a polymap $s : \Psi, A, \Psi' \mid \Gamma \longrightarrow \Delta$ in $\mathscr{X}$. Composing (on A) results in a polymap in $\mathscr{X}$ of type

$$f \cdot s : \Psi, \Phi, \Psi' \mid \Gamma \longrightarrow \Delta.$$

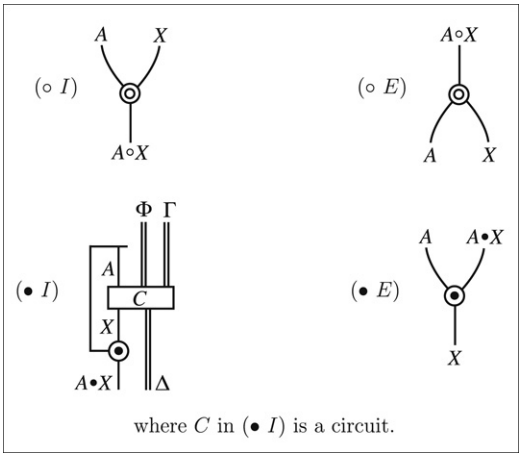
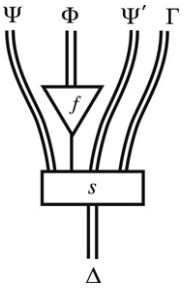


Fig. 7. ∘ and • circuit introduction and elimination rules.

where the centre dot notation "$\cdot$" represents cutting a multimap into a polymap. In circuit notation this is represented as

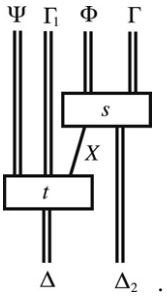


.

The second type of composition is between polymaps in $\mathscr{X}$. Suppose $s : \Phi \mid \Gamma \longrightarrow \Delta_1, X, \Delta_2$ and $t : \Psi \mid \Gamma_1, X, \Gamma_2 \longrightarrow \Delta$ are polymaps in $\mathscr{X}$. Composing (on X) results in a polymap in $\mathscr{X}$ of type

$$s\,;\, t : \Phi, \Psi \mid \Gamma_1, \Gamma, \Gamma_2 \longrightarrow \Delta_1, \Delta, \Delta_2.$$

One possible simplified circuit diagram (in which $\Gamma_2 = \Delta_1 = \emptyset$ so that there are no crossings) for this cut is given by



.

All of the evident associativity and interchange laws must, of course, hold in any poly-actegory.

There are circuits corresponding to introduction and elimination rules for each of the connectives $\otimes$, $\oplus$, $\top$, and $\bot$. We refer the interested reader to [10] for the full story. Here we are interested in the action rules ∘ and •. The introduction and elimination circuit diagrams for ∘, which we label by $(\circ\, I)$ and $(\circ\, E)$, are given in Fig. 7. Notice the similarity to the introduction and elimination rules of the $\otimes$ and $\oplus$ connectives [10].

The • rule is a binding rule in the sense that the introduction rule must involve a "scope box" [14] so that they are only applicable to the situation where one has a subcircuit C to attach the link to. It replaces a derivation $A, \Phi \mid \Gamma \Vdash X, \Delta$ with a derivation $\Phi \mid \Gamma \Vdash A \bullet X, \Delta$. The circuit diagrams, labelled by $(\bullet\, I)$ and $(\bullet\, E)$, are also given in Fig. 7. With this rule notice the similarity to the introduction and elimination rules (including the scope box) for $\multimap$, the linear implication [14].

For any connective there are two types of circuit rewrites: *reductions* which allow one to simplify a circuit which involves an elimination rule immediately after an introduction rule, and *expansions* which "split" a wire carrying a compound formula into "simpler" wires, and ultimately to atomic wires. The reductions and expansions for ∘ and • are given by in Fig. 8.

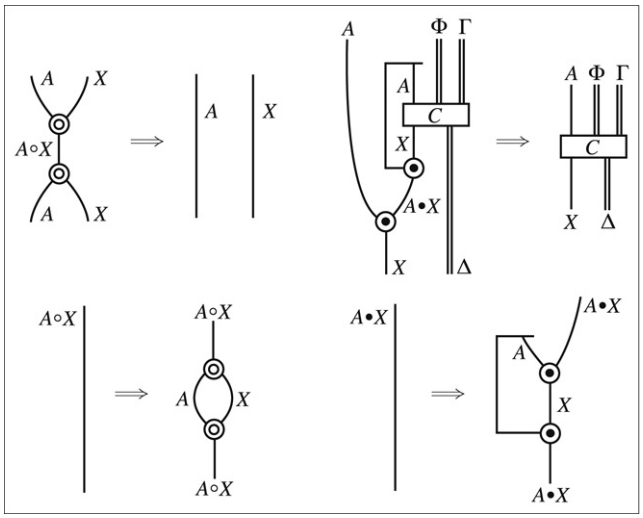


Fig. 8. Circuit reduction and expansion rules.

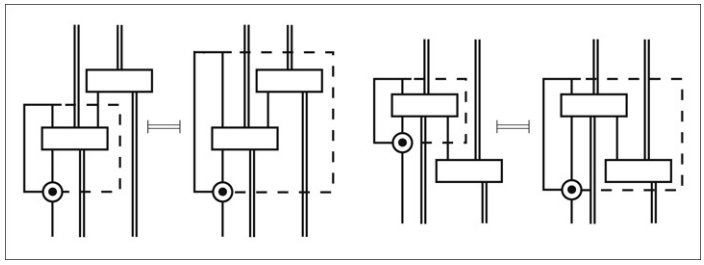
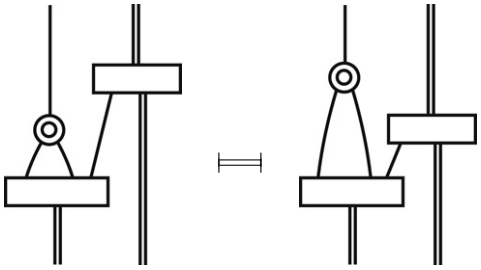


Fig. 9. • scope rules.

In circuits, links are allowed to slide up or down corresponding to cut elimination steps or equations. For example, in simplified circuit notation, sliding of the ∘ node



corresponds to the [sequent-$\circ_l$] cut-elimination rule. This should be familiar, however, the scoping rules for • may not be. We present the equations (where we have again slightly simplified the circuit notation) in Fig. 9.

6.2. Representability of the actions

There are natural bijections

$$\frac{\Phi, A \mid \Gamma, X \Vdash \Delta}{\Phi \mid \Gamma, A \circ X \Vdash \Delta} \quad \text{and} \quad \frac{\Phi, A \mid \Gamma \Vdash Y, \Delta}{\Phi \mid \Gamma \Vdash A \bullet Y, \Delta}$$

obtained in the top-to-bottom direction respectively by the $\circ_l$ and $\bullet_r$ inference rules, and in the bottom-to-top direction by cutting respectively with the derivations

$$\frac{\dfrac{\mathrm{id}_A}{A \vdash A} \quad \dfrac{\mathrm{id}_X}{\emptyset \mid X \Vdash X}}{A \mid X \Vdash A \circ X} \quad \text{and} \quad \frac{\dfrac{\mathrm{id}_A}{A \vdash A} \quad \dfrac{\mathrm{id}_X}{\emptyset \mid X \Vdash X}}{A \mid A \bullet X \Vdash X}.$$

These are called the *representing polymaps* for the covariant and contravariant actions respectively. They are clearly natural as when represented they are the identity maps.

The representing polymap of the covariant action may be described as a term:

$$\alpha[x] \cdot \gamma =_X \alpha \;\; :: \;\; x : A \mid \gamma : X \Vdash \alpha : A \circ X.$$

which allows one to see the bijection at the level of the term logic. Suppose

$$s :: \Phi, x : A \mid \Gamma, \alpha : X \Vdash \Delta.$$

The calculation

$$\begin{aligned} \alpha[x] \cdot \gamma =_X \alpha \;;_\alpha \alpha\langle y\rangle \cdot s &\Longrightarrow \gamma =_X \alpha \;;_\alpha s \\ &\Longrightarrow s, \end{aligned}$$

where the first rewrite is the $[\circ_r\text{-}\circ_l]$ cut-elimination step, establishes one direction of the bijection. Now suppose

$$t :: \Phi \mid \Gamma, \alpha : A \circ X \Vdash \Delta.$$

The other direction is given by

$$\begin{aligned} \gamma\langle x\rangle \cdot (\alpha[x] \cdot \gamma =_X \alpha \;;_\alpha t) &\Longrightarrow (\gamma\langle x\rangle \cdot \alpha[x] \cdot \gamma =_X \alpha) \;;_\alpha t \\ &\Longrightarrow \gamma =_{A\circ X} \alpha \;;_\alpha t \\ &\Longrightarrow t \end{aligned}$$

where the first rewrite is the [$\circ_l$-sequent] cut-elimination step in reverse, and the second uses that $\gamma\langle x\rangle \cdot \alpha[x] \cdot \gamma =_X \alpha$ is the identity on $A \circ X$.

6.3. Representability for poly-actegories

The proof theory of the message-passing logic is a poly-actegory. In this setting each polymap, besides having multiple inputs and outputs can, in addition, have inputs from a multicategorical world. Although the inputs may be typed to come from different worlds, once this distinction is erased one is simply left with a polycategory in which certain types are served by multimaps alone. This view determines the requirements on the poly-actegory composition. Demanding representability of the (multicategorical) tensor $*$ and its unit I, the (polycategorical) tensor $\otimes$ and its unit $\top$, the par $\oplus$ and its unit $\bot$, the covariant action $\circ$, and the contravariant action $\bullet$, then forces, we claim, all the proof equivalences discussed in Section 3.4 for the message-passing logic.

To prove this would be stretching the patience of the reader and is, besides, relatively standard categorical proof theory. Instead we shall focus on how a linear actegory arises from the maps of these settings. Given the circuit representation it is actually very straightforward to verify that the required coherence diagrams are satisfied. Thus, the main objective of this section is to show how the data of a linear actegory arises.

A significant feature of a linear actegory is the parameterised adjunction between the covariant and contravariant action. This arises directly from the representability by the following two-way series of inferences:

$$\dfrac{\dfrac{\dfrac{A \circ X \Vdash Y}{A \mid X \Vdash Y}}{X \Vdash A \bullet Y}}$$

We can also derive all the coherence isomorphisms using representability. Here we give the derivation of such for the binary connectives (the units coherences are derived in a similar manner):

$$a^*_\circ \;\; : \;\; \dfrac{\dfrac{\dfrac{\dfrac{(A * B) \circ X \Vdash Y}{A * B \mid X \Vdash Y}}{A, B \mid X \Vdash Y}}{A \mid B \circ X \Vdash Y}}{A \circ (B \circ X) \Vdash Y} \qquad a^*_\bullet \;\; : \;\; \dfrac{\dfrac{\dfrac{\dfrac{X \Vdash (A * B) \bullet Y}{A * B \mid X \Vdash Y}}{A, B \mid X \Vdash Y}}{A \mid X \Vdash B \bullet Y}}{X \Vdash A \bullet (B \bullet Y)}$$

$$a^\circ_\otimes \;\; : \;\; \dfrac{\dfrac{\dfrac{\dfrac{A \circ (X \otimes Y) \Vdash Z}{A \mid X \otimes Y \Vdash Z}}{A \mid X, Y \Vdash Z}}{A \circ X, Y \Vdash Z}}{(A \circ X) \otimes Y \Vdash Z} \qquad a^\bullet_\oplus \;\; : \;\; \dfrac{\dfrac{\dfrac{\dfrac{X \Vdash A \bullet (Y \oplus Z)}{A \mid X \Vdash Y \oplus Z}}{A \mid X \Vdash Y, Z}}{X \Vdash A \bullet Y, Z}}{X \Vdash (A \bullet X) \oplus Z}$$

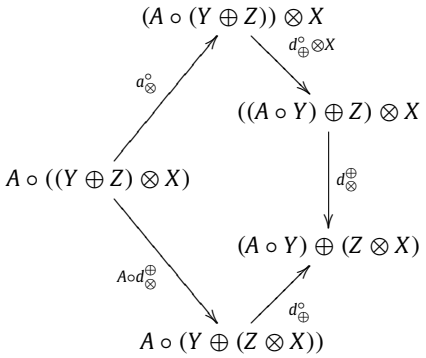
To derive the distributions we have to use the poly-actegorical composition together with the representing polymaps. The derivations are below, however, note that these are not derivations of the logic, but in a representable poly-actegory.

In this setting we have just the poly-actegorical composition, the representing polymaps, and the equivalences. The binary inference in each of the inferences below is the poly-actegorical composition.

$$d^{\circ}_{\oplus}:\ \dfrac{\dfrac{X\oplus Y \Vdash X, Y \qquad A \mid X \Vdash A\circ X}{A \mid X\oplus Y \Vdash A\circ X, Y}}{A\circ(X\oplus Y)\Vdash (A\circ X)\oplus Y} \qquad d^{\bullet}_{\otimes}:\ \dfrac{\dfrac{A \mid A\bullet X \Vdash X \qquad X, Y \Vdash X\otimes Y}{A \mid A\bullet X, Y \Vdash X\otimes Y}}{(A\bullet X)\otimes Y \Vdash A\circ(X\otimes Y)}$$

$$d^{\circ}_{\bullet}:\ \dfrac{\dfrac{B \mid B\bullet X \Vdash X \qquad A \mid X \Vdash A\circ X}{A, B \mid B\bullet X \Vdash A\circ X}}{A\circ(B\bullet X)\Vdash B\bullet(A\circ X)}$$

To prove soundness of the message passing logic we have presented a recipe which relies on the fact (established in the next section) that to be the category of maps of a representable poly-actegory is precisely to be a linear actegory (Theorem 6.1). To check that the proof equivalences of the message passing logic will hold in a representable poly-actegory is then straightforward. As an example, consider the coherence diagram (19) which we recall here:



It is an easy circuit calculation, which we show in Figs. 10 and 11, to see that both routes of the coherence diagram are equivalent.

We conclude:

Proposition 6.2. *The category of maps of a representable poly-actegory form a linear actegory.*

6.4. Soundness

This section is devoted to showing that all the reductions and equations in a representable poly-actegory $\mathscr{P}$ (e.g., $\mathscr{P} = \mathbf{PMsg}$) hold in any linear actegory $\mathscr{X}$. That is, we wish to show that given a linear actegory one may build from it a representable poly-actegory. Once again we concentrate on the action rules and refer the reader to the start of [12] for a description of representability, and to [15], for soundness for representable polycategories. For simplicity, we will ignore most instances of associativity.

The first step is to show how the two cut rules of a poly-actegory arise from the data of a linear actegory. The "action cut" of a multimap into a polymap will be described first. To this end suppose that we have a multimap and polymap

$$f : \Phi \longrightarrow A \qquad \text{and} \qquad s : A, \Psi \mid \Gamma_1, X, \Gamma_2 \longrightarrow \Delta$$

respectively that are represented by the maps (between singletons)

$$\widetilde{f} : \widetilde{\Phi} \longrightarrow A \qquad \text{and} \qquad \widetilde{s} : \widetilde{\Gamma}_1 \otimes A\circ X \otimes \widetilde{\Gamma}_2 \longrightarrow \widetilde{\Delta}$$

in $\mathscr{X}$ (the tildes over Γ's and Φ's indicating the results of representing). The composite of f and s on A should be a polymap in $\mathscr{P}$ as

$$f\cdot s : \Phi, \Psi \mid \Gamma_1, X, \Gamma_2 \longrightarrow \Delta. \tag{X}$$

In $\mathscr{X}$ we may form the composite

$$\widetilde{\Gamma}_1 \otimes \widetilde{\Phi}\circ X \otimes \widetilde{\Gamma}_2 \xrightarrow{1\otimes\widetilde{f}\circ X\otimes 1} \widetilde{\Gamma}_1 \otimes A\circ X \otimes \widetilde{\Gamma}_2 \xrightarrow{\widetilde{s}} \widetilde{\Delta}$$

which, by reversing representability, gives the desired polymap (X).

The composite of two polymaps can be defined in a similar manner. Suppose that

$$s : \Phi \mid \Gamma \longrightarrow \Delta_1, X, \Delta_2 \qquad \text{and} \qquad t : \Psi \mid \Gamma_1, X, \Gamma_2 \longrightarrow \Delta$$

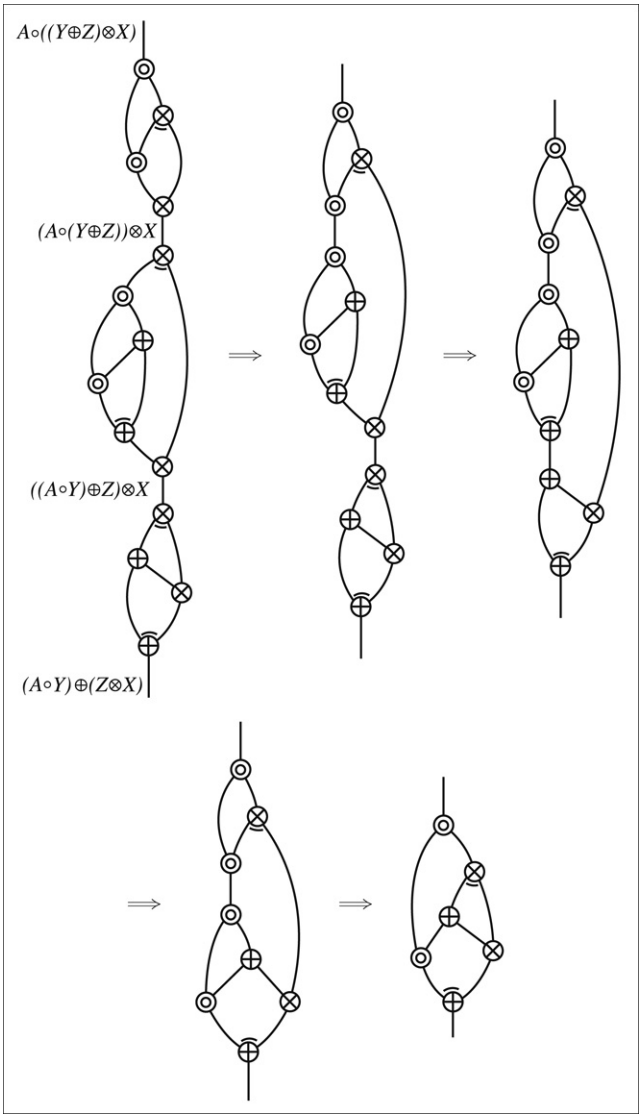


Fig. 10. Circuit coherence calculation: top route.

are polymaps represented by

$$\widetilde{s} : \widetilde{\Gamma} \longrightarrow (\widetilde{\Delta}_1 \oplus X) \oplus \widetilde{\Delta}_2 \qquad \text{and} \qquad \widetilde{t} : \widetilde{\Gamma}_1 \otimes (X \otimes \widetilde{\Gamma}_2) \longrightarrow \widetilde{\Delta} \,.$$

The composite of s and t on X should be a polymap in $\mathscr{P}$ as

$$s; t : \Phi, \Psi \mid \Gamma_1, \Gamma, \Gamma_2 \longrightarrow \Delta_1, \Delta, \Delta_2. \tag{Y}$$

In $\mathscr{X}$ we may form the composite

$$\widetilde{\Gamma}_1 \otimes \widetilde{\Gamma} \otimes \widetilde{\Gamma}_2 \xrightarrow{1\otimes\widetilde{s}\otimes 1} \widetilde{\Gamma}_1 \otimes ((\widetilde{\Delta}_1 \oplus X) \oplus \widetilde{\Delta}_2) \otimes \widetilde{\Gamma}_2$$

$$\xrightarrow{1\otimes d_{\otimes}^{\oplus'}} \widetilde{\Gamma}_1 \otimes ((\widetilde{\Delta}_1 \oplus X) \otimes \widetilde{\Gamma}_2) \oplus \widetilde{\Delta}_2 \xrightarrow{1\otimes d_{\otimes}^{\oplus}\oplus 1} \widetilde{\Gamma}_1 \otimes (\widetilde{\Delta}_1 \oplus (X \otimes \widetilde{\Gamma}_2)) \oplus \widetilde{\Delta}_2$$

$$\xrightarrow{d_{\oplus}^{\otimes'}\oplus 1} \widetilde{\Delta}_1 \oplus (\widetilde{\Gamma}_1 \otimes (X \otimes \widetilde{\Gamma}_2)) \oplus \widetilde{\Delta}_2 \xrightarrow{1\oplus\widetilde{t}\oplus 1} \widetilde{\Delta}_1 \oplus \widetilde{\Delta} \oplus \widetilde{\Delta}_2$$

which, by reversing representability, gives the desired polymap (Y).

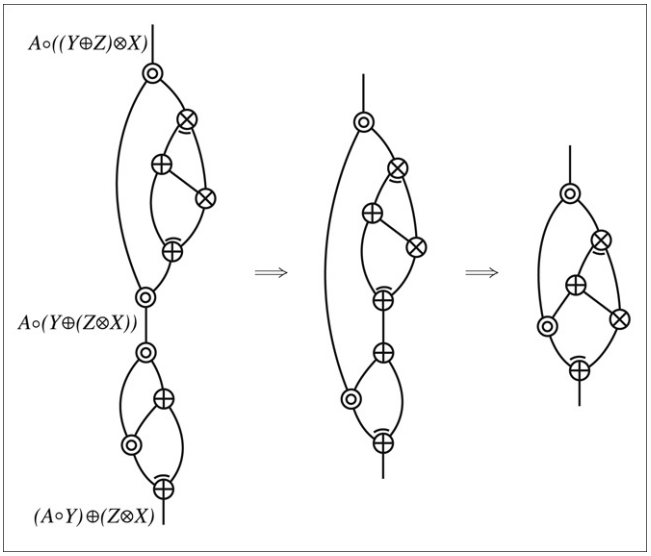


Fig. 11. Circuit coherence calculation: bottom route.

This shows how the two cuts arise in the representable poly-actegory built from a linear actegory. Associativity and the interchange laws follow from a combination of functoriality, naturality, and associativity in $\mathscr{X}$. As an example suppose there are multimaps and a polymap

$$f : \Phi \longrightarrow A,\ \ f' : \Phi' \longrightarrow A',\ \ \text{and}\ \ s : \Psi_1, A, \Psi_2, A', \Psi_3 \mid \Gamma, X, \Gamma_2, X', \Gamma_3 \longrightarrow \Delta$$

which are represented as

$$\widetilde{f} : \widetilde{\Phi} \longrightarrow A,\quad \widetilde{f}' : \widetilde{\Phi}' \longrightarrow A',\quad \text{and}\quad \widetilde{s} : \widetilde{\Gamma} \otimes A \circ X \otimes \widetilde{\Gamma}_2 \otimes A' \circ X' \otimes \widetilde{\Gamma}_3 \longrightarrow \widetilde{\Delta}.$$

If the interchange law is to hold then

$$f \cdot (f' \cdot s) = f' \cdot (f \cdot s).$$

From the definition the left-hand side and right-hand side respectively are given by reversing representability in the composites below.

$$\begin{array}{c}
\widetilde{\Gamma} \otimes \widetilde{\Phi} \circ X \otimes \widetilde{\Gamma}_2 \otimes \widetilde{\Phi}' \circ X' \otimes \widetilde{\Gamma}_3 \\
\downarrow{\scriptstyle 1\otimes\widetilde{f}\circ X\otimes 1\otimes 1\otimes 1} \\
\widetilde{\Gamma} \otimes A \circ X \otimes \widetilde{\Gamma}_2 \otimes \widetilde{\Phi}' \circ X' \otimes \widetilde{\Gamma}_3 \\
\downarrow{\scriptstyle 1\otimes 1\otimes 1\otimes\widetilde{f}'\circ X'\otimes 1} \\
\widetilde{\Gamma} \otimes A \circ X \otimes \widetilde{\Gamma}_2 \otimes A' \circ X' \otimes \widetilde{\Gamma}_3 \\
\downarrow{\scriptstyle \widetilde{s}} \\
\widetilde{\Delta}
\end{array}
\qquad
\begin{array}{c}
\widetilde{\Gamma} \otimes \widetilde{\Phi} \circ X \otimes \widetilde{\Gamma}_2 \otimes \widetilde{\Phi}' \circ X' \otimes \widetilde{\Gamma}_3 \\
\downarrow{\scriptstyle 1\otimes 1\otimes 1\otimes\widetilde{f}'\circ X'\otimes 1} \\
\widetilde{\Gamma} \otimes \widetilde{\Phi} \circ X \otimes \widetilde{\Gamma}_2 \otimes A' \circ X' \otimes \widetilde{\Gamma}_3 \\
\downarrow{\scriptstyle 1\otimes\widetilde{f}\circ X\otimes 1\otimes 1\otimes 1} \\
\widetilde{\Gamma} \otimes A \circ X \otimes \widetilde{\Gamma}_2 \otimes A' \circ X' \otimes \widetilde{\Gamma}_3 \\
\downarrow{\scriptstyle \widetilde{s}} \\
\widetilde{\Delta}
\end{array}$$

That both composites are equal follows from functoriality of $\otimes$.

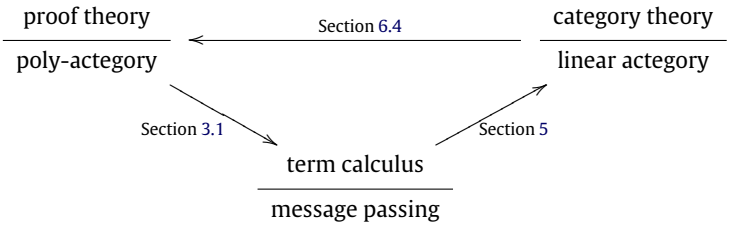
Proposition 6.3. *Every linear actegory is the category of maps of some representable poly-actegory.*

This now proves Theorem 6.1, that to be a linear actegory is precisely to be the category of maps of a representable poly-actegory.

Coproducts in the polycategorical setting are discussed extensively in [24]. They have not been mentioned the discussion of this section as their presence or absence is completely orthogonal to the main result.

7. Conclusion

We have now completed the tour of the diagram connecting proof theory, categorical semantics, and term calculus as promised:



A reasonable question to ask is whether we really have a complete set of coherence diagrams and of equations. Our response to this is, of course, that we did try to be reasonably complete. However, we are happy to admit that, in a system of this size, it is quite possible that we have overlooked something. That said, however, as we now have at least three different ways to view the subject matter all of which agree, we are confident that the basic story is complete and that these ideas, insofar as they are not completely fleshed-out here, can be.

The aim of the paper was to show that message-passing could be accommodated in the proof theoretic framework for concurrency provided by the two-sided proof theory of cut-elimination (which is a fragment of linear logic). In particular we feel that by providing a categorical semantics we have anchored this correspondence in a way which will facilitate the exploration of semantic models. Thus, we feel that we have now lain out the story of how message passing can be modeled proof theoretically and categorically in sufficient detail to establish the viability of this perspective. There remains a lot to be done.

In the process of writing the paper the proof system underwent a number of downsizing changes in an attempt to make it more manageable. For example, the additives at the message passing level were sacrificed for this reason. From the programming perspective, features such as (the initial and final) datatypes are desirable *at both levels* and this is an aspect to which we would like to return. Particularly, at the message-passing level datatypes are of significant interest as they allow the expression of communication protocols in a formal manner.

References

[1] S. Abramsky, Computational interpretations of linear logic, Theoret. Comput. Sci. 111 (1–2) (1993) 3–57.
[2] S. Abramsky, Interaction categories (extended abstract), in: G.L. Burn, S.J. Gay, M.D. Ryan (Eds.), Proc.of 1st Imperial College Dept. of Computing Wksh. on Theory and Formal Methods (Chelwood Gate, March 1993), in: Workshops in Computing, Springer, London, 1993, pp. 57–69.
[3] S. Abramsky, Interaction categories and communicating sequential processes, in: A.W. Roscoe (Ed.), A Classical Mind: Essays in Honour of C.A.R. Hoare, Prentice Hall, New York, 1994, pp. 1–16.
[4] S. Abramsky, S. Gay, R. Nagarajan, Specification structures and propositions-as-types for concurrency, in: G. Birtwistle, F. Moller (Eds.), Logics for Concurrency: Structure versus Automata (Proc. of 8th Higher-Order Wksh., Aug./Sept. 1995), in: Lect. Notes in Comput. Sci., vol. 1043, Springer, Berlin, 1996, pp. 5–40.
[5] S. Abramsky, P.-A. Melliès, Concurrent games and full completeness, in: Proc. of 14th Ann. IEEE Symp. on Logic in Computer Science, LICS' 99, Trento, July 1999, IEEE CS Press, Los Alamitos, CA, 1999, pp. 431–442.
[6] A. Barber, P. Gardner, M. Hasegawa, G. Plotkin, From action calculi to linear logic, in: Selected Papers from 11th Int. Wksh. on Computer Science Logic, CSL' 97, Aarhus, Aug. 1997, in: Lect. Notes in Comput. Sci., vol. 1414, Springer, Berlin, 1998, pp. 78–97.
[7] H.P. Barendregt, The Lambda Calculus: Its Syntax and Semantics, Revised ed., in: Studies in Logic and the Foundations of Mathematics, vol. 103, North-Holland, Amsterdam, 1984.
[8] G. Bellin, P.J. Scott, On the π-calculus and linear logic, Theoret. Comput. Sci. 135 (1) (1994) 11–65.
[9] N. Benton, A mixed linear and non-linear logic: Proofs, terms and models (extended abstract), in: L. Pacholski, J. Tiuryn (Eds.), Selected Papers from 8th Int. Wksh. on Computer Science Logic, CSL' 94, Kazimierz, Sept. 1994, in: Lect. Notes in Comput. Sci., vol. 933, Springer, Berlin, 1995, pp. 121–135.
[10] R.F. Blute, J.R.B. Cockett, R.A.G. Seely, T.H. Trimble, Natural deduction and coherence for weakly distributive categories, J. Pure Appl. Algebra 133 (3) (1996) 229–296.
[11] L. Cardelli, A.D. Gordon, Mobile ambients, in: M. Nivat (Ed.), Proc. of 1st Int. Conf. on Foundations of Software Science and Computation Structure, FoSSaCS' 98, Lisbon, March/Apr. 1998, in: Lect. Notes in Comput. Sci., vol. 1378, Springer, Berlin, 1998, pp. 140–155.
[12] J.R.B. Cockett, J. Koslowski, R.A.G. Seely, Morphisms and modules for poly-bicategories, Theory Appl. Categ. 11 (2003) 15–74.
[13] J.R.B. Cockett, C.A. Pastro, A language for multiplicative-additive linear logic, in: L. Birkedal (Ed.), Proc. of 10th Conf. on Category Theory and Computer Science, CTCS 2004, Copenhagen, Aug. 2004, in: Electron. Notes in Theor. Comput. Sci., vol. 122, Elsevier, Amsterdam, 2005, pp. 23–65.
[14] J.R.B. Cockett, R.A.G. Seely, Proof theory for full intuitionistic linear logic, bilinear logic, and mix categories, Theory Appl. Categ. 3 (1997) 85–131.
[15] J.R.B. Cockett, R.A.G. Seely, Weakly distributive categories, J. Pure Appl. Algebra 114 (2) (1997) 133–173. Corrected version at http://www.math.mcgill.ca/rags/.
[16] C. Hermida, Representable multicategories, Adv. Math. 151 (2) (2000) 164–225.
[17] D.J.D. Hughes, R.J. van Glabbeek, Proof nets for unit-free multiplicative-additive linear logic, ACM Trans. Comput. Log. 6 (4) (2005) 784–842.
[18] C.B. Jay, Languages for monoidal categories, J. Pure Appl. Algebra 59 (1) (1989) 61–85.
[19] J. Lambek, Deductive systems and categories II: Standard constructions and closed categories, in: Category Theory, Homology Theory and Their Applications, in: Lect. Notes in Math., vol. 86, Springer, Berlin, 1969, pp. 76–122.
[20] P. McCrudden, Categories of representations of balanced coalgebroids, Ph.D. Thesis, Macquarie University, 1999.
[21] I. Mackie, L. Román, S. Abramsky, An internal language for autonomous categories, Appl. Categ. Structures 1 (3) (1993) 311–343.
[22] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes I, Inform. and Comput. 100 (1) (1992) 1–40.
[23] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes II, Inform. and Comput. 100 (1) (1992) 41–77.
[24] C. Pastro, $\Sigma\pi$-polycategories additive linear logic process semantics, Master's Thesis, University of Calgary, 2004.